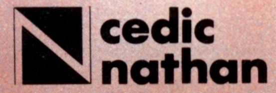
ALICE
ALICE 90
VOTRE MICRO-ORDINATEUR









ALICE, ALICE-90

VOTRE MICRO-ORDINATEUR

Jean Delcourt



dirigée par Serge Pouts-Lajus



Dans la même collection

Minitel, votre guide pratique
Jacques David

ZX-Spectrum, votre micro-ordinateur
Serge Pouts-Lajus

Oric-Atmos, votre micro-ordinateur
Michel Bussac

MO5, votre micro-ordinateur
Serge Pouts-Lajus

Dictionnaire micro-informatique
Éric Duceau/Christophe Doë

Portatifs, les nouveaux micro-ordinateurs
Pierre Raguenes/Gérard Sitbon

MO5, TO7, TO7-70, vos programmes
Pierre Champeaux/Serge Pouts-Lajus

Oric-Atmos, vos programmes
Michel Bussac/Gill Espèche

La micro en 100 questions
Bruno Delatour

Macintosh, votre micro-ordinateur
Jean-Baptiste Touchard

Choisir son micro-ordinateur
Yvon Dargery

Commodore 64, votre micro-ordinateur
Daniel Isidore

Apple //c, votre micro-ordinateur
Éric Duceau/Christophe Doë

BASIC, votre langage de programmation
Serge Pouts-Lajus

Logo, votre langage de programmation
André Myx et Pierre Subtil

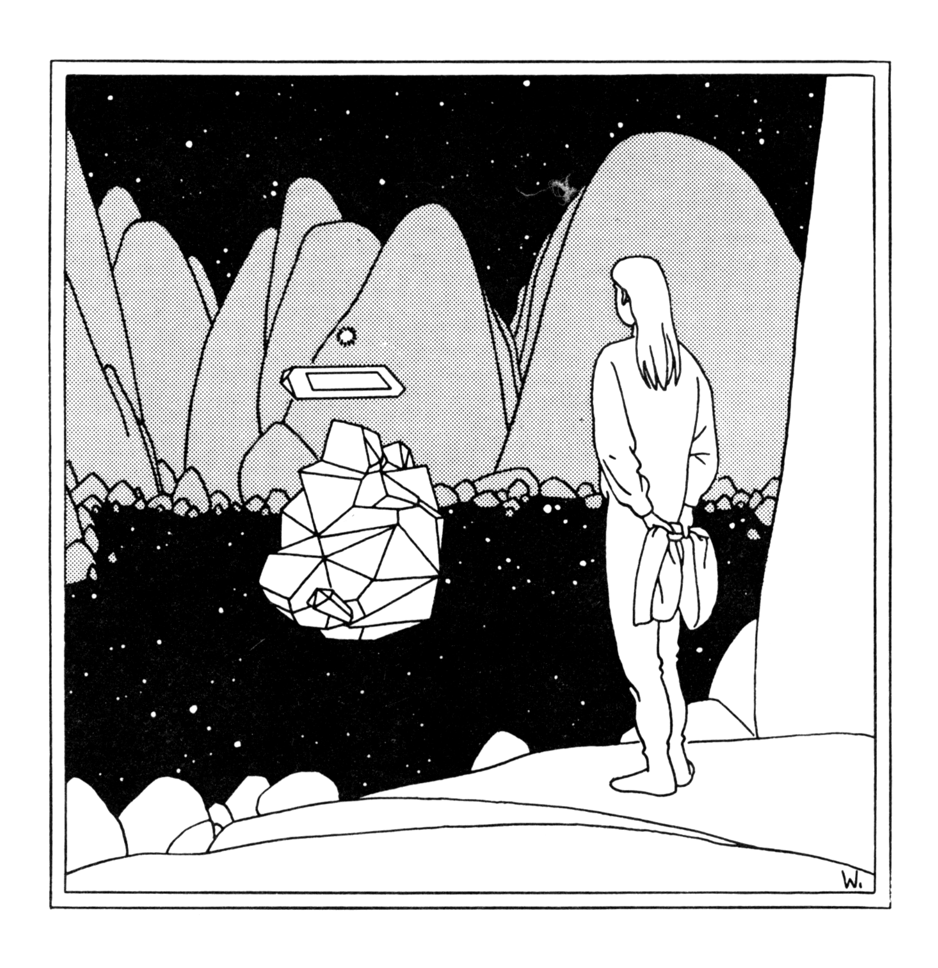
Couverture : Studio Bercy   Photos : Fabien Robineau
Illustrations : Walter Lalonde, Laetita Boucrot
Maquette : Véronique Chabert d'Hières   Montage : Berthe Nevière

Ce volume porte la référence
ISBN 2-7124-1515-9

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS

SOMMAIRE

INTRODUCTION



Alice est apparu dans le monde des micro-ordinateurs fin 1983 : du matériel français, à un prix abordable, le succès n'a pas tardé.

Aujourd'hui, la famille d'Alice s'agrandit avec deux nouveautés :

• Une version musclée d'Alice : plus de mémoire, plus de capacité (accès à un éditeur-assembleur).

• Alice-90 : encore plus de mémoire, les mêmes possibilités mais dans une carrosserie métallique aux lignes futuristes... Et un vrai clavier.

Une fois n'est pas coutume, ces machines sont compatibles ; les programmes déjà écrits pour le premier Alice vont « tourner » sur les autres.

Ainsi, l'ambition de MATRA, faire de l'informatique familiale « une merveille de simplicité », trouve un nouvel élan.

DESCRIPTION



ALICE, PREMIER CONTACT

Plusieurs présentations possibles :

• Une présentation ordinaire pour Alice-90 : on trouve dans un emballage en carton (à l'abri des chocs néanmoins) le micro-ordinateur, les manuels, l'alimentation et le câble de raccordement au récepteur TV. A vous de vous procurer un magnétophone et le câble adéquat si vous voulez conservez vos programmes.

• Une version «coffret» pour Alice ou bien pour Alice-90 : une élégante mallette en plastique rouge renferme micro-ordinateur, manuels et câbles mais aussi quelques logiciels et un magnétophone; assurément, une présentation efficace et agréable. Ce n'est pas seulement une question d'esthétique : Alice est ainsi un authentique ordinateur *transportable*. Vous pourrez l'emporter en week-end ou chez vos amis. Vous noterez au passage la grande subtilité du vocabulaire informatique :

— un ordinateur *portable* dispose d'une alimentation électrique autonome; il est possible de l'utiliser dans le métro (surtout si on a une place assise);

— un ordinateur *transportable* est alimenté par le secteur et nécessite souvent (c'est le cas d'Alice) un dispositif de visualisation (TV); mais il peut être transporté aisément par une seule personne.

Quelle que soit la présentation, vous disposerez de deux manuels :

— un livre d'initiation au BASIC, destiné aux utilisateurs débutants, et qui se présente sous la forme d'un guide de programmation méthodique. Abondamment illustré, contenant de nombreux exercices corrigés, il permet au fabricant de dire qu'Alice est le premier «livre-ordinateur».

— un fascicule de présentation de l'éditeur-assembleur, qui se contente de décrire les instructions sans donner ni beaucoup d'explications ni d'exemples; le débutant devra chercher ses sources ailleurs...

Examinons maintenant l'ordinateur lui-même.

• La version «Alice» est la copie conforme du premier Alice : boîtier en plastique rouge de dimensions réduites (5 cm x 18 cm x 22 cm), de poids 850 g.

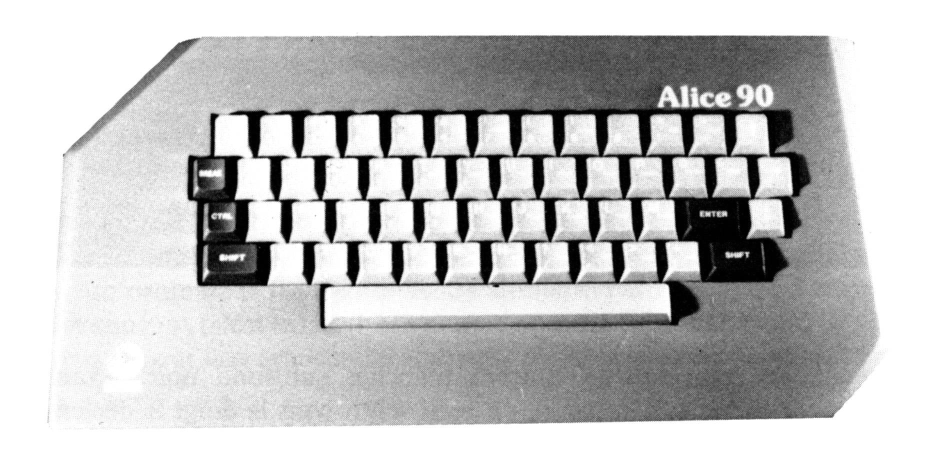


Le clavier comporte 48 touches blanches sur fond noir. Avant même de brancher l'appareil, on peut sentir sous le doigt le déclenchement des touches et prévoir que la frappe sera aisée : pas de risque de rebonds, pas de fatigue ; les touches ne sont pas en gomme comme sur beaucoup de micro-ordinateurs bas de gamme ; elles font penser aux touches d'une calculatrice.

La qualité du clavier d'un micro-ordinateur est un atout important. C'est hélas à ce niveau que les constructeurs sont souvent négligents. Le clavier est la partie mécanique du micro-ordinateur et un bon clavier coûte cher ; l'obligation de «serrer» les prix contraint souvent les constructeurs à proposer des claviers de qualité médiocre. Cet inconvénient n'apparaît qu'à l'usage ; il faut donc être attentif à ce détail au moment de l'achat.

Incontestablement, le constructeur d'Alice a veillé à la qualité de son clavier.

Aucune comparaison, pourtant, avec le clavier d'une bonne machine à écrire car les touches sont petites et rapprochées. Mais en ce qui concerne la frappe des programmes qui ne se fait jamais à un rythme frénétique, le clavier d'Alice nous a semblé tout à fait satisfaisant, d'autant plus que les doigts des programmeurs risquent d'être le plus souvent de petits doigts...



• Alice-90 fait, dès l'abord, une autre impression : d'un volume plus important, sa carrosserie métallique a un côté «professionnel»; sans toutefois faire illusion : le design très étudié et très original n'a pas sa place dans un trop austère bureau. Par contre, Alice-90 gardera un certain standing au salon familial, à côté de la chaîne hi-fi et d'un magnétoscope.

Le clavier d'Alice-90 est un «vrai» clavier, style machine à écrire. Il comporte quelques touches supplémentaires, par rapport à Alice, mais nous aurons l'occasion d'y revenir;

Dans les deux cas, les touches semblent avoir des fonctions multiples puisqu'elles peuvent porter jusqu'à trois lettres ou symboles; la manipulation du clavier demandera certainement un temps d'apprentissage, et les fausses manoeuvres devraient être fréquentes au début.

Mise en marche

Alice est un micro-ordinateur d'initiation. Il est donc destiné à des utilisateurs novices. Chacun sait qu'un ordinateur c'est au moins :

— un clavier,
— un écran.

Pour le clavier, pas de problème, il est là.
Pour l'écran, il faudra utiliser votre propre téléviseur à condition qu'il soit équipé d'une prise PERITEL. Cette prise standard est obligatoire sur tous les appareils construits après 1980.

Si votre téléviseur ne dispose pas de la prise PERITEL, vous pourrez vous brancher sur la prise antenne moyennant l'achat d'un adaptateur spécial assez coûteux. Dans ces conditions, l'image qui vous sera délivrée sera en noir et blanc, même si votre poste est un poste couleur.

Le jeu des couleurs sur Alice étant l'un de ses principaux attraits, cette solution semble donc peu recommandable. Il faut savoir que *tous* les micro-ordinateurs familiaux (appelés aussi nano-ordinateurs) proposés sur le marché depuis 1981 génèrent des images en couleur. Il paraît raisonnable de penser que pour les usagers de ces machines, l'incitation à se procurer des téléviseurs récents est forte et justifiée.

Si donc vous possédez un poste de télévision couleur, il est raisonnable de penser que vous avez aussi l'électricité. Tout comme le grille-pain et le séchoir à cheveux, Alice est un appareil électrique.

Mais la tension nécessaire à son fonctionnement n'est que de 10 V. Le transformateur convertit donc le courant de 220 V en un courant de 10 V. Il est branché d'une part à la prise marquée ALIMENTATION à l'arrière du micro-ordinateur, d'autre part à une prise femelle ordinaire. Pourquoi un transformateur extérieur à l'ordinateur? D'abord pour une question de place, mais surtout pour une question de chaleur : la transformation du courant produit une énergie thermique qui peut être importante après une longue utilisation; dans un petit boîtier comme celui d'Alice, cette chaleur pourrait causer d'importants dégâts aux fragiles circuits électroniques.

Pour connecter Alice au téléviseur, il faut utiliser le second câble et le brancher d'une part à la prise marquée PRISE-TV à l'arrière du micro-ordinateur, d'autre part à la prise PERITEL du téléviseur. Attention, dans le cas d'Alice-90, ce câble comporte aux deux extrémités une prise PERITEL mâle : la prise marquée d'une bague rouge doit être connectée au récepteur TV, l'autre à l'arrière d'Alice.

Ces manoeuvres simples étant effectuées, il suffit d'allumer d'abord le téléviseur. Si vous vous trouvez sur l'une des trois chaînes officielles de la Télévision Française, à l'heure du journal télévisé... le présentateur habituel vous apparaîtra comme à l'accoutumée. Mais si de plus vous basculez l'interrupteur marche/arrêt d'Alice, vous devriez voir ceci :

```
MICROCOLOR BASIC 1.0

COPYRIGHT 1982 MICROSOFT

OK
```

Si rien de tout cela ne se passe, ne courez pas tout de suite chez votre revendeur : il arrive souvent que les prises diverses soient mal enfoncées, surtout quand le matériel est neuf. Signalons enfin que la prise PERITEL peut être laissée en permanence branchée sur le téléviseur. C'est d'ailleurs préférable, car une telle prise avec de nombreux contacts est relativement fragile.

Interface

Ce premier chapitre avait pour but de faire sentir au lecteur que grâce à l'évolution des techniques, l'odinateur est devenu un objet domestique d'usage simple. La mise en marche d'Alice ne pose pas plus de problèmes que celle d'un grille-pain.

Il en va tout autrement en ce qui concerne *l'utilisation* de ce nouvel objet dans l'environnement domestique.

Osons faire une comparaison avec un autre objet devenu familier : l'automobile.

Que ce soit au sujet de l'ordinateur ou de l'automobile, les deux grandes questions auxquelles doit répondre le futur utilisateur sont celles-ci :

— à quoi ça sert?
— comment ça marche?

En ce qui concerne la première de ces deux questions, qui est effectivement une question cruciale (si la réponse devait être «à rien», les conclusions seraient rapidement tirées), les réactions face à l'automobile et face à l'ordinateur sont très différentes.

Tout le monde connaît les services que peut rendre une automobile : voyager, garder le chien pendant qu'on fait les courses, etc. En revanche, à la question «à quoi sert un ordinateur familial?», la réponse mérite quelques développements...

Il nous semble juste, en ce qui concerne l'odinateur familial, de commencer par la seconde question : comment ça marche?

Même s'il n'est pas nécessaire de dominer parfaitement la mécanique d'une automobile pour la conduire, quelques caractéristiques techniques sont connues de tous :

— sous le capot, il y a un moteur à essence;
— le volant permet de tourner les roues;
— il existe un réservoir d'essence;
— il existe un circuit électrique alimenté par une batterie.

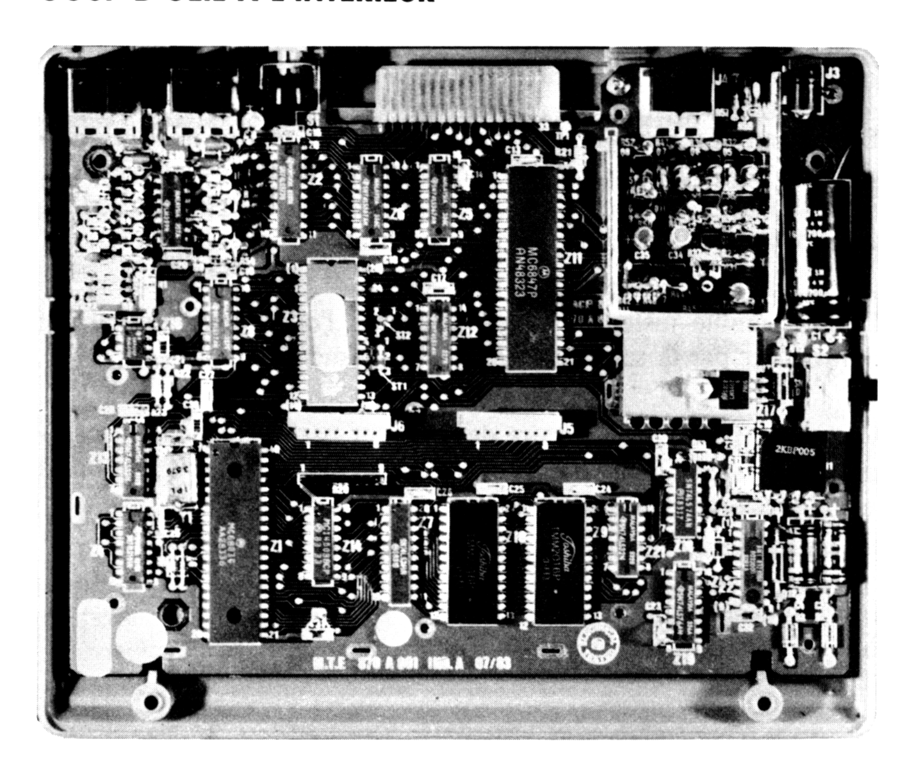
Il n'est pas certain qu'il en aille de même avec un micro-ordinateur. C'est un objet trop neuf dans l'environnement pour qu'on puisse être certain que les principes techniques de son fonctionnement soient connus de tous.

Lorsqu'on achète une automobile, il n'est pas absurde de soulever le capot, de regarder comment le moteur tourne. Nous allons donc soulever le capot d'Alice...

SOUS LE CLAVIER D'ALICE

Quatre vis seulement permettent de démonter le boîtier d'Alice et d'accéder à la partie électronique. Mauvaise surprise, l'une d'entre elles est protégée (par un adhésif ou un cachet de cire suivant la version) : comme l'indique le bon, ouvrir l'appareil annulera la garantie. Peut-être n'aimeriez-vous pas courir un tel risque ; nous l'avons fait pour vous.

COUP D'OEIL A L'INTERIEUR



Le couvercle enlevé, voici donc l'âme de la machine révélée dans sa nudité et sa complexité.

Au départ, on peut être rassuré : tout semble bien rangé. Sur une plaque verte (on dit plutôt une «carte») sont alignés des petits dominos noirs à pattes : ce sont les *puces*.

Du pied de chaque patte, ou presque, partent des fils argentés incrustés dans la carte. Ils relient les dominos suivant une logique qui nous dépasse encore mais que nous allons tenter d'éclairer quelque peu.

De ci, de là, on reconnait des composants familiers à qui s'est déjà risqué à désosser un appareil radio : diodes, résistances, condensateurs.

Ce n'est qu'au moment où l'ordinateur est branché que tout le système prend vie : bien sûr, les puces ne se mettent pas à sauter sur place mais il se produit entre elles de fantastiques échanges d'informations sous la forme de signaux électriques.

LES PUCES

Les puces, reliées les unes aux autres par les bus, ont chacune un rôle à tenir dans le grand ballet des octets. Elles ont des physionomies assez différentes : certaines sont plus petites que d'autres, certaines ont seize pattes, d'autres quarante.

Mais les différences les plus importantes ne peuvent se déceler à l'oeil nu : elles proviennent de la spécificité de leurs rôles.

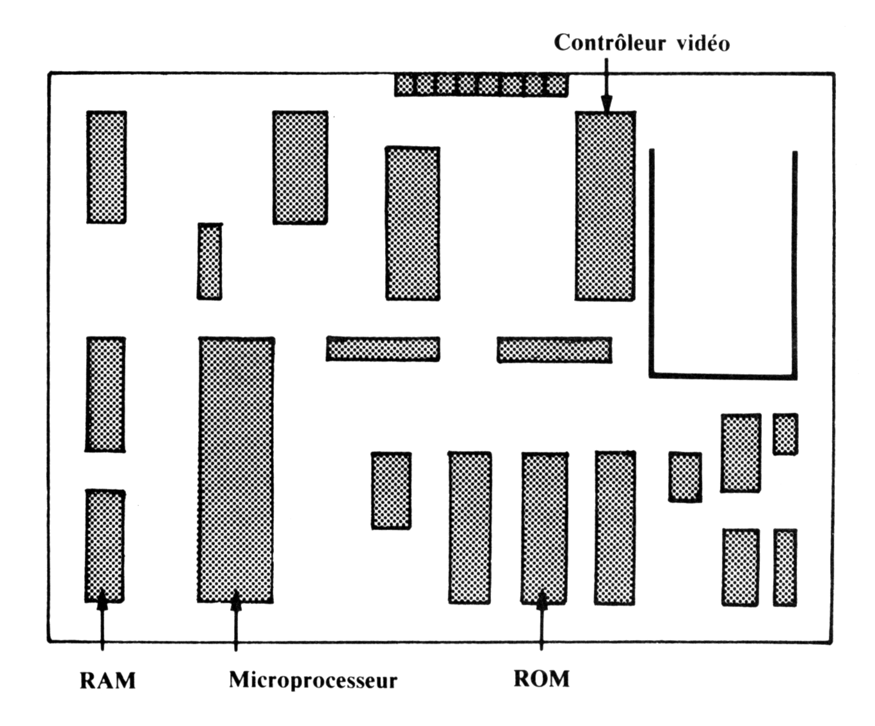
Il existe trois catégories de puces :

— *Les puces mémoires* dont le rôle est de stocker l'information pour un usage à long ou à court terme.

— *Les puces d'interfaçage* qui traitent l'information en provenance de l'extérieur (par exemple le clavier) pour la mettre sous forme utilisable.

— La reine des puces : *le microprocesseur*, imposant par sa taille (quarante pattes), sa renommée, sa puissance de travail, sa vélocité, et qui est de fait et à juste titre, le véritable cerveau de l'ordinateur.

Regardez, sur la figure, la répartition des différentes puces.



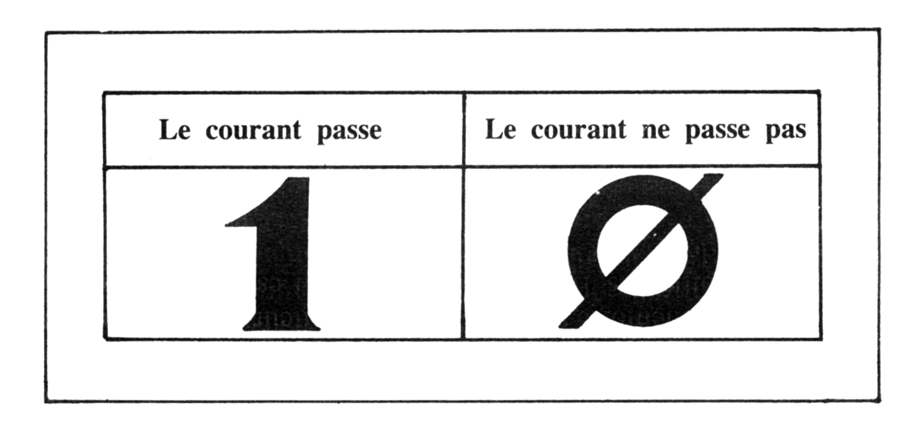
LES BUS

Ainsi sont nommés les fils qui relient les puces entre elles. Il s'agit tout simplement de conducteurs métalliques, mais plutôt que d'utiliser de classiques fils électriques, le constructeur a préféré les dessiner dans la carte elle-même.

Concrètement, les passagers de ces bus sont des électrons, c'est-à-dire de l'électricité. Ce courant électrique est utilisé pour le transport d'informations (puisqu'il s'agit d'informatique) suivant des règles de base d'une extrême simplicité.

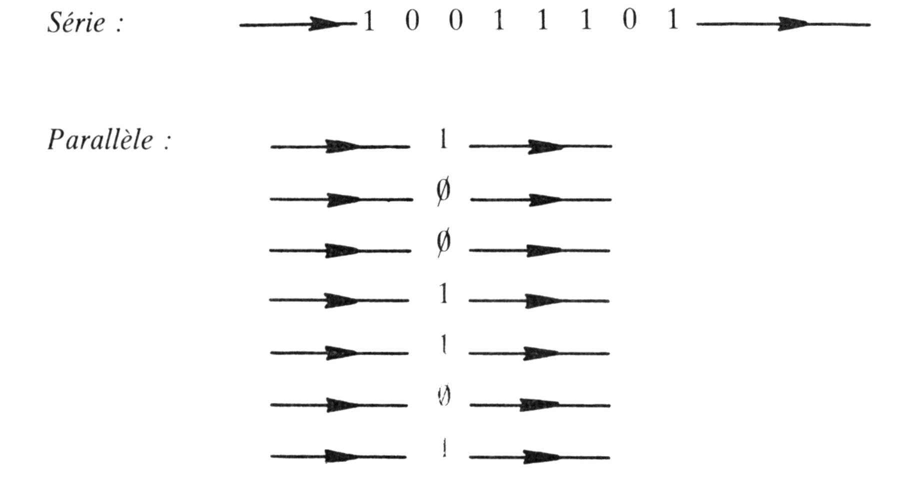
Ce morceau si élémentaire d'information, équivalent à OUI ou NON, s'appelle un BIT. Il paraît difficile d'imaginer une conversation soutenue entre deux personnes qui s'interdiraient d'utiliser d'autre mots que OUI ou NON. Pourtant, pensez à l'alphabet Morse : à l'aide d'informations tout aussi élémentaires (un trait, un

point), on pouvait transmettre des messages, lettres, mots, phrases entières. Tout est affaire de codage. Le codage utilisé dans les micro-ordinateurs est assez complexe dans ses détails mais il est possible de s'en faire une idée.



Les bits sont toujours regroupés par paquets de huit : un octet.

Si les huit bits d'un octet circulent à la queue leu leu dans un seul fil, on dit qu'il s'agit d'une transmission «en série». Si les huit bits d'un octet circulent ensemble dans huit fils électriques, on dit qu'il s'agit d'une transmission « en parallèle ».



Nous verrons plus loin que ce regroupement des bits en octets permet d'utiliser un codage quasi-universel en informatique : le code ASCII, analogue à l'alphabet Morse.

LES DEUX MEMOIRES

Ce qui distingue en premier lieu les deux mémoires, c'est une différence de capacité.

La capacité d'une mémoire se mesure avec une unité spéciale aux informaticiens : le kilo octet (ou Ko, ou K). Un kilo octet, comme son nom ne l'indique pas exactement, est formé de 1024 octets. Pourquoi 1024 et pas 1000? Parcequ'en informatique, le nombre deux est roi à cause de ce point de départ obligé : 0 et 1, ça fait 2.

Bits et octets



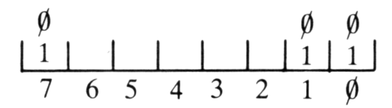
Un bit est la plus petite cellule d'information qu'un ordinateur peut traiter.

1
|Ø|

Ici, 0 et 1 représentent sous une forme mathématique le fait qu'un signal électrique est transmis (1) ou ne l'est pas (0). Chaque cellule a donc deux représentations possibles.

1 1
|Ø|Ø|

Un ensemble de deux cellules présente quatre combinaisons.



Un octet est un ensemble de huit bits, soit 2^8 combinaisons possibles (256). Chaque octet peut représenter un nombre entre 0 et 255.

Un octet est formé de 8 bits ($8 = 2 x 2 x 2$) et un kilo octet est formé de 1024 octets car : $1024 = 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 = 2^{10}$

Si une mémoire a une capacité de 16K, elle peut donc stocker 16 x 1 024 = 16 384 octets, soit 131 072 unités élémentaires d'information.

Considérons, pour fixer les idées, qu'un octet suffit pour mémoriser une lettre ou un chiffre. Une mémoire de 16 Ko permettrait de mémoriser entièrement 16 pages de ce livre.

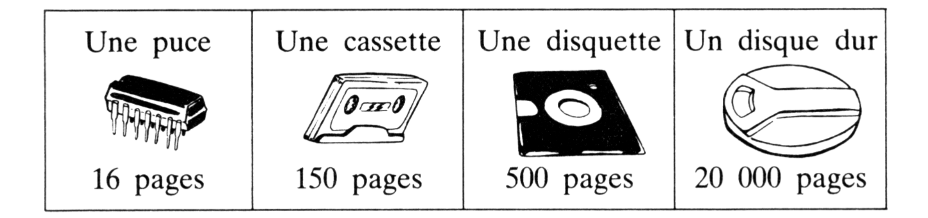
Avouons-le : c'est un peu décevant.

A partir de là, il est facile de comprendre pourquoi il n'est pas souhaitable de stocker toutes les informations dans la mémoire de l'ordinateur : il y aurait rapidement saturation, embouteillages...

C'est pourquoi, on utilise des mémoires *externes* pour les informations dont on n'a pas besoin constamment : bandes magnétiques, disquettes, disques durs.

Les disques durs, par exemple, ont une capacité mémoire qui peut atteindre une vingtaine de Méga-octets ou *million* d'octets.

Capacité mémoire



Une puce	Une cassette	Une disquette	Un disque dur
16 pages	150 pages	500 pages	20 000 pages

MEMOIRE MORTE, MEMOIRE VIVE

En dehors de leurs capacités, les puces mémoire se distinguent par leur accès : certaines mémoires sont «figées», elles ont emmagasiné des octets une fois pour toutes et il n'est plus question d'y changer quoi que ce soit.

Il est cependant possible de les consulter, d'aller y chercher des informations : c'est le plus souvent le microprocesseur qui s'en charge.

Ces mémoires sont les mémoires mortes : ce sont en quelque sorte les dictionnaires de l'ordinateur, imprimés une fois pour toutes. En anglais on dit, et en français on le dit aussi : ROM, Read Only Memories.

D'autres mémoires ont un comportement plus souple que celui des mémoires mortes : ce sont les mémoires vives. En anglais, RAM, d'après Random Access Memories.

Les mémoires vives sont les pages blanches sur lesquelles le microprocesseur (encore lui), peut lire, écrire ou modifier les octets.

Au point de vue technologique, cette différence se révèle importante : les mémoires mortes sont des circuits intégrés figés qui garderont leurs caractéristiques, donc leurs informations, même quand l'alimentation sera coupée. Au contraire, les mémoires vives sont «volatiles» et l'information qu'elles contiennent a fortement tendance à s'échapper : il faut constamment les rafraîchir par un peu d'électricité.

Dès que l'on éteint l'ordinateur, la mémoire vive se vide, la mémoire morte reste intacte.

MEMOIRES VIVES ET MEMOIRES MORTES : A QUOI SERVENT-ELLES ?

C'est décidé, nous disons :
RAM pour mémoire vive
ROM pour mémoire morte
(à l'envers, ROM, ça fait MOR...)

Les RAM paraissent plus souples d'emploi, plus agréables, plus intéressantes parce qu'on peut y écrire et y effacer : elles jouent le rôle d'un tableau noir.

Mais les ROM sont indispensables; si elles n'existaient pas l'ordinateur serait, chaque fois qu'on le branche, un cerveau complètement vide, incapable de comprendre ce qu'on tenterait de lui expliquer en tapant frénétiquement sur le clavier. C'est pourquoi tout micro-ordinateur dispose de ROM qui contiennent au moins les informations nécessaires à la mise en route, à la gestion des organes de communication (clavier, écran). Les ROM peuvent également contenir le «langage évolué» (BASIC par exemple) qui servira à l'utilisateur pour écrire ses programmes.

C'est donc dans la RAM que se trouvent les programmes : de la capacité de la RAM dépend donc la taille des programmes exécutables par la machine. Il est facile de comprendre que pour conserver un programme avant d'éteindre l'ordinateur, il est nécessaire de le transporter, le sauvegarder sur une mémoire extérieure : cassette ou disquette.

Il ne faudrait pourtant pas croire que toute la RAM soit disponible à l'utilisateur : une partie est réservée à l'ordinateur lui-même; il y met les données variables qui lui sont utiles; c'est une sorte de cahier de brouillon. On appelle souvent cette partie de la mémoire vive : RAM système. Une autre partie, la RAM vidéo contient toutes les informations nécessaires à l'affichage sur l'écran.

Parlons chiffres maintenant.

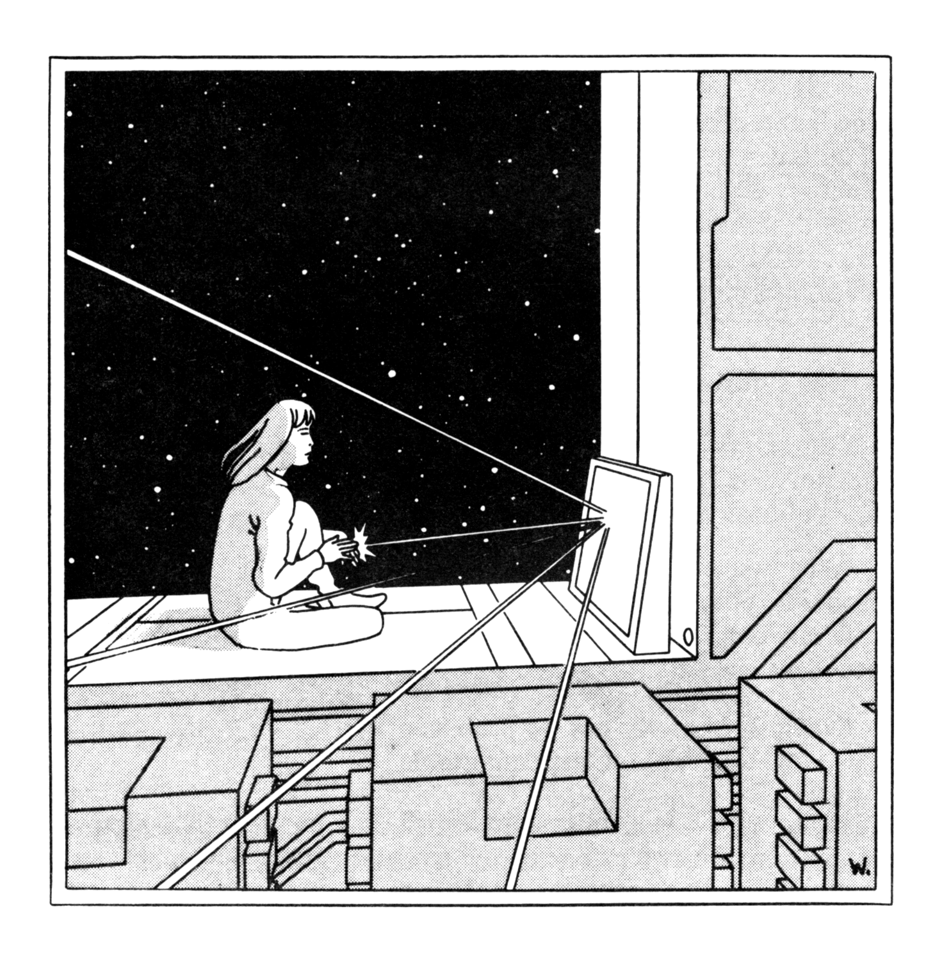
— Le premier Alice permettait de disposer d'une RAM de 4 Ko, dont un peu plus des trois-quarts étaient réellement disponibles pour l'utilisateur : c'est bien peu.

— Alice nouvelle version offre 16 Ko de RAM, dont la moitié est disponible pour l'utilisateur; la RAM vidéo est en effet plus importante que dans le cas précédent, à cause d'une définition graphique nettement accrue.

— Quant à Alice-90, pas de problème : 40 Ko de RAM, dont 32 pour vous tout seul... C'est confortable.

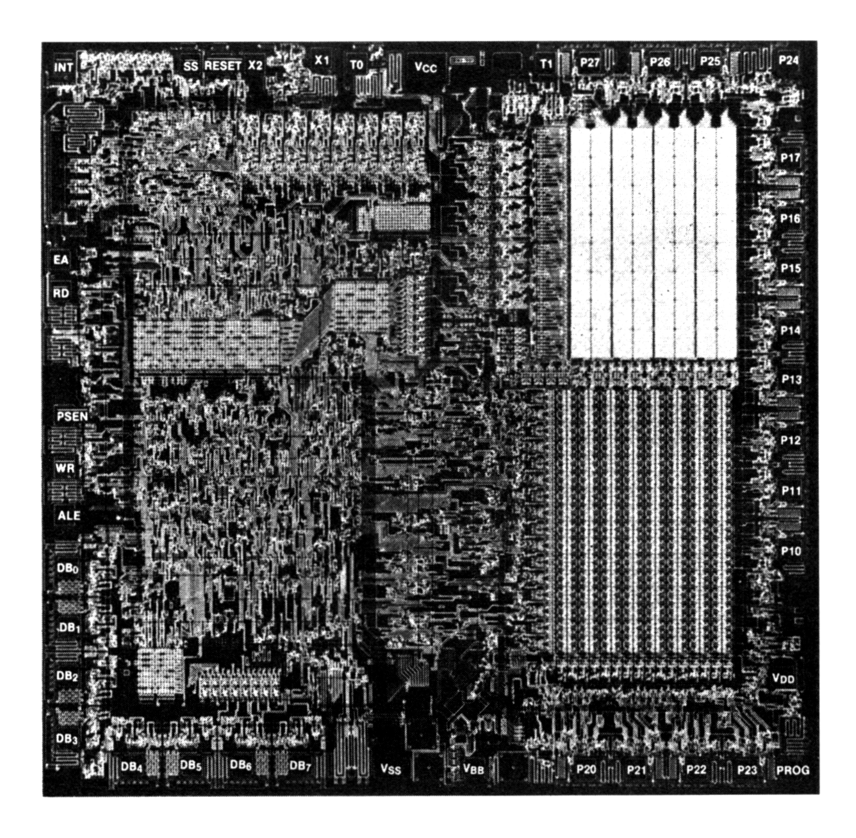
Signalons que la capacité mémoire d'Alice peut être étendue, en ajoutant un boîtier d'extension mémoire; on atteint alors 24 Ko utilisateur, se rapprochant ainsi d'Alice-90.

La ROM, qui contient les langages BASIC et ASSEMBLEUR, ainsi que l'EDITEUR est de 16 Ko ; deux fois plus que dans la première version d'Alice qui, c'est vrai, n'offrait que le BASIC.

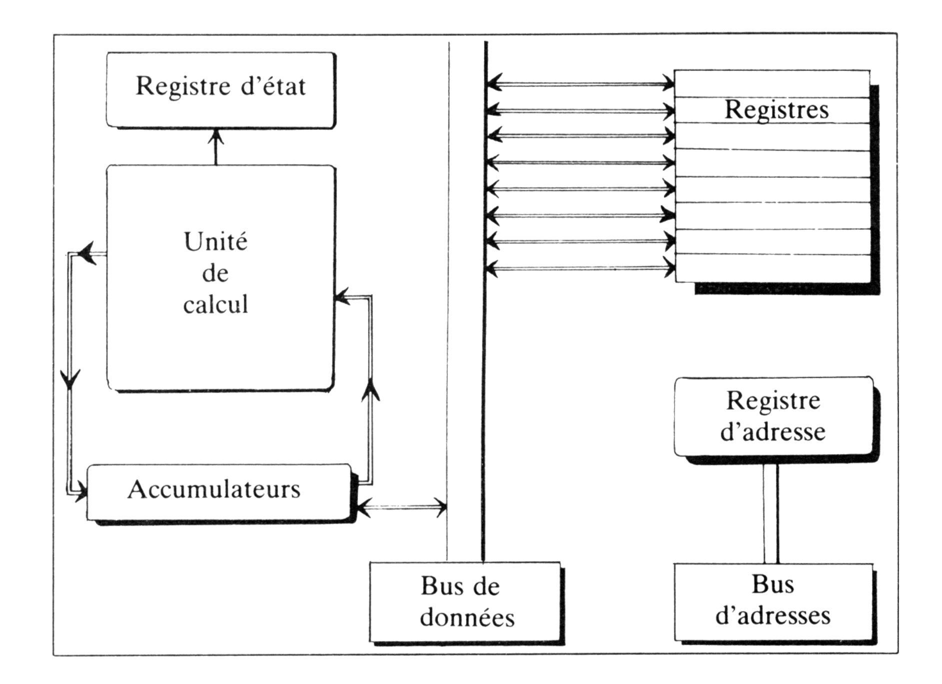


LE ROI MICROPROCESSEUR

Le microprocesseur est le roi, l'âme, le cerveau, le chef d'orchestre de l'ordinateur. C'est lui qui dirige entièrement la manoeuvre ; c'est un véritable miracle de la technologie d'avoir réussi à enfermer tant de puissance dans une si petite boîte. Il est donc normal que cette puce-là soit un peu plus compliquée que les autres. Il n'est pourtant pas nécessaire de connaître son fonctionnement pour pouvoir utiliser l'ordinateur. Heureusement, car voici un agrandissement d'un microprocesseur :



Pourtant, et à condition de ne pas rentrer dans trop de détails, l'architecture générale d'un microprocesseur est très éclairante pour qui désire s'initier à l'informatique.



Ce schéma décrit une partie seulement de l'activité du microprocesseur et c'est déjà assez compliqué. Cette partie, *l'unité de calcul*, fonctionne comme une mini-calculatrice. Elle peut effectuer des opérations : additions, soustractions, etc.

Les nombres sur lesquels ces opérations sont effectuées peuvent être pris par l'unité de calcul :
— dans les *accumulateurs*,
— dans les *registres* qui sont des mémoires propres au microprocesseur,
— dans les mémoires RAM et ROM par l'intermédiaire du bus de données.

Le *registre d'état* est une mémoire «pense-bête», aide-mémoire. Son rôle est, entre autres, de comptabiliser les retenues, de repérer les résultats nuls ce qui sera bien utile pour tester l'égalité de deux nombres.

En s'enfonçant encore un peu plus profondément dans la logique du microprocesseur, nous parvenons jusqu'à *l'unité de commande* qui est à elle seule le «cerveau» de toute l'opération.

L'unité de commande est le grand organisateur :
— elle décode les instructions reçues ;
— elle organise l'exécution de ces instructions. C'est elle qui oriente le passage des données entre les registres, la mémoire et l'unité de calcul.
— Elle met à jour le registre d'adresses qui indique à quel endroit de la mémoire, il faut aller lire ou écrire un nombre.

Le travail de l'unité de commande est mené à un rythme d'enfer, imposé par une minuscule horloge à quartz qui, telle un métronome, organise le séquencement des multiples tâches. Particularité de ce métronome : il bat environ 4 000 000 fois par seconde !

Le microprocesseur est un 6803, récent rejeton d'une famille célèbre, la famille « 6800 », lancée par la firme américaine MOTOROLA. Sans rentrer dans trop de détails techniques, signalons simplement que c'est un microprocesseur *8 bits*, c'est-à-dire qui traite des octets. Son *espace d'adressage* est de 64 Ko : cela signifie qu'il peut gérer 65 536 cases mémoires différentes. Ces caractéristiques n'ont rien d'original. Ce sont celles de tous les microprocesseurs utilisés dans les ordinateurs familiaux. On trouve maintenant, mais pour des matériels « haut de gamme », des microprocesseurs 16 bits, voire 32 bits.

LES PUCES D'INTERFACES

Ces puces qui ne sont ni des mémoires, ni le microprocesseur, peuvent avoir un rôle tout à fait marginal. Un rôle de « tampon » par exemple, c'est-à-dire de stockage temporaire des données (Buffer en anglais).

D'autres au contraire, ont un rôle primordial : elles permettent à l'ordinateur de ne pas être un schizophrène absolu, complètement refermé sur lui-même.

Les puces d'interfaces servent *d'interprètes* pour toutes les communications venant de l'extérieur, ou dans l'autre sens de *traductrices* pour tout envoi d'informations hors de l'ordinateur. Ainsi, c'est une

puce spécialisée qui traduira l'appui d'une touche sur le clavier en un code interprétable par le microprocesseur. C'est une autre puce spécialisée qui traduira les résultats d'une opération en commandes envoyées au téléviseur pour qu'il affiche ce résultat à l'écran.

Les puces d'interfaces envoient au microprocesseur des signaux spéciaux, les *interruptions*, pour qu'il interrompe le travail courant et se mette à l'écoute de l'extérieur.

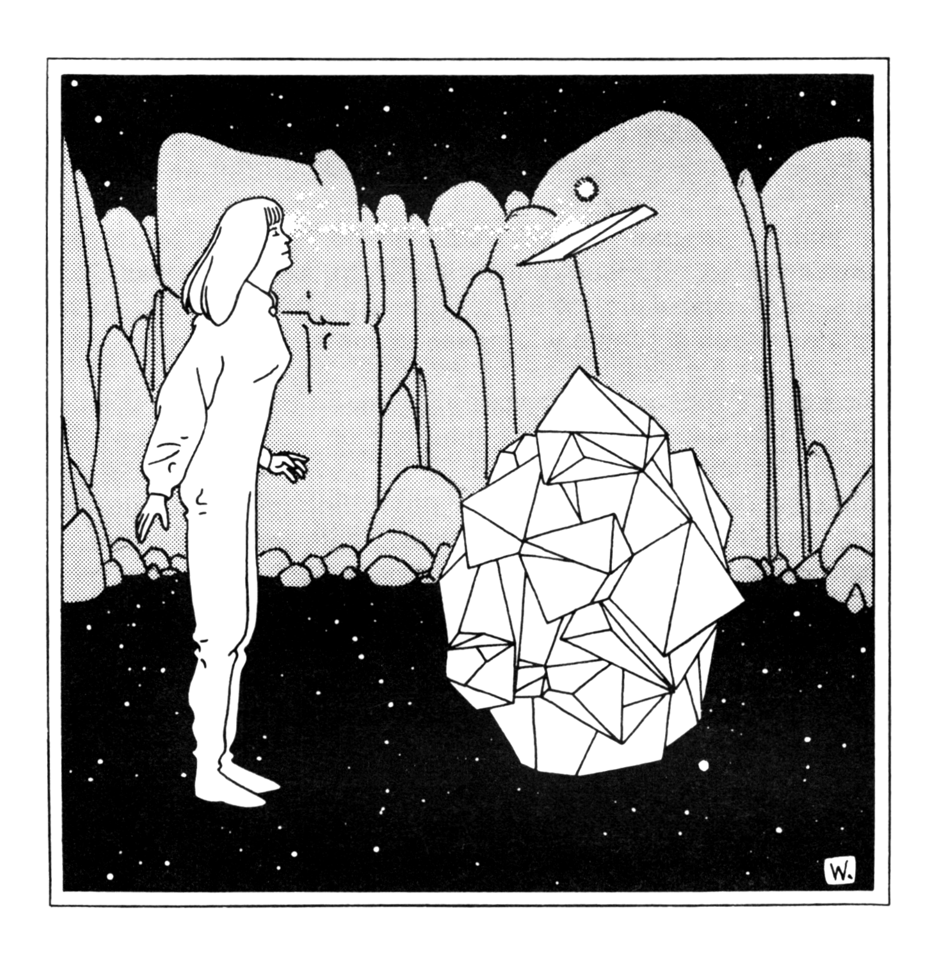
POUR EN FINIR AVEC LA TECHNIQUE

Ce rapide survol des entrailles du micro-ordinateur nous a permis de comprendre comment tout est simple au départ et si compliqué à l'arrivée. Chaque organe visité sur la carte du microprocesseur est l'aboutissement d'un immense progrès technologique et mériterait à lui seul de longs développements.

Avant de refermer le capot de notre machine, n'oublions pas que sous ce fragile capot, c'est une usine en miniature qui s'affaire. Certains exécutants sont spécialisés. Leurs circuits intégrés ne sont disponibles que pour une tâche bien précise : mémorisation, traduction. Un seul en fait est polyvalent, c'est le microprocesseur : il organise toute l'activité de l'usine, réception, fabrication, stockage, livraison. Il est à tout moment disponible pour toute tâche qu'on voudrait lui voir exécuter (sauf préparer du café...).

C'est cette totale disponibilité du microprocesseur qui en fixe la limite : rien à en tirer si on ne lui donne pas d'ordres, si on ne le nourrit pas d'un programme.
C'est pourquoi l'homme, créateur et utilisateur du microprocesseur doit intervenir pour nourrir cette puce insatiable.

PROGRAMMATION EN BASIC



LANGAGE MACHINE ET LANGAGES EVOLUES

Dans le chapitre précédent, nous avons exploré la partie matérielle de l'ordinateur : description de l'aspect extérieur, détail des composants électroniques. C'est à peu près ce qu'on appelle le « Hardware ».

Après le corps, l'esprit. Les informaticiens qui connaissent les langues parlerons de « Software ». Nous parlerons plutôt de « logiciel » : l'ensemble des programmes qui permettent d'utiliser l'ordinateur.

Pour commencer, tout peut se décrire en termes de dialogue entre l'utilisateur et l'ordinateur; qui dit dialogue dit langage; ce sont les langages qui sont les programmes les plus importants d'un ordinateur.

Le premier langage, le plus rudimentaire, s'appelle le *langage machine* : c'est celui que connaît le microprocesseur, le seul d'ailleurs. Son vocabulaire est limité, cela dépend un peu des microprocesseurs; en général une centaine de mots, parfois un peu plus.

Que peut-on dire avec ce langage machine? Certainement pas des phrases très élaborées, ou très subtiles : on ne s'adresse qu'à un microprocesseur, et le langage machine servira à lui donner des ordres ou *instructions* qu'il devra suivre fidèlement. Les instructions que comprend le microprocesseur sont peu nombreuses; il s'agit essentiellement :

— d'instructions de *transfert* du contenu d'une case mémoire dans une autre, avec éventuellement l'intervention des registres qui sont, rappelons-le, les mémoires propres au microprocesseur;

— d'instructions de *calcul* portant sur les contenus des registres ou cases mémoires. Tous les microprocesseurs savent faire des additions, soustractions, multiplications par deux. Certains en connaissent un peu plus, mais n'oublions pas que les cases mémoires des

micro-ordinateurs ne contiennent qu'un octet, qui ne peut coder plus de 256 nombres. Toutes les opérations que sait faire le microprocesseur ne porteront donc que sur de «petits» nombres. Le 6803 sait, par exemple, multiplier deux entiers codés sur chacun un octet; le résultat doit alors être codé sur deux octets.

— d'instructions de *branchement* : en règle générale, le microprocesseur exécute les instructions dans l'ordre; on dit «séquentiellement». Mais il peut arriver que l'on désire rompre cet ordre. On pourra alors demander au microprocesseur de faire un «saut» de tant de pas dans le programme. Ces instructions de branchement sont en général associées à des instructions de *test* : on parle alors de branchements conditionnels.

Comme vous l'avez compris, une suite d'instructions constitue un programme. Le langage machine sert donc à communiquer des programmes au microprocesseur. Mais, très rudimentaire, il lui faudra beaucoup de mots pour exprimer des choses très simples.

Penons un exemple. Pour effectuer l'addition de deux entiers codés sur un octet, il faudra :

— transférer le premier opérande dans l'accumulateur A;

— additionner le contenu de A au second opérande;

— tenir compte d'une éventuelle retenue (en examinant le registre d'état);

— enfin, transférer le résultat de l'opération dans une case mémoire, en vue d'une utilisation ultérieure.

Le langage machine est d'une prolixité redoutable, mais c'est la langue maternelle du microprocesseur : il la comprend parfaitement, et exécute quasi instantanément (quelques millionièmes de seconde) chaque instruction.

Les programmes de jeux rapides, par exemple, sont tous écrits en langage machine.

Le moniteur-assembleur

Nous n'avons pas encore compris, concrètement, comment on peut communiquer avec le microprocesseur; il faut utiliser le langage machine mais avec quel véhicule?

Voilà comment cela se passe : dès la mise sous tension, le microprocesseur examine le contenu d'une case mémoire. Souvent celle d'adresse Ø, parfois celle d'adresse 65 535. Il doit y trouver une instruction, qu'il exécute, puis il passe à la suivante. Il faut donc que dès la mise sous tension il y ait un programme dans les cases mémoires : c'est ce qu'on appelle le *moniteur*. Il est en mémoire ROM, et c'est lui, par exemple, qui s'occupe de l'affichage sur l'écran de votre téléviseur, ou scrute le clavier pour déceler l'appui d'une touche. Certains moniteurs permettent aussi de converser avec le microprocesseur : on peut alors directement écrire des programmes en langage machine; les instructions seront des octets, écrits sous codage binaire (des 0 et des 1) ou sous codage hexadécimal, plus condensé.

Le moniteur d'Alice ne donne pas cette possibilité; il nous met tout de suite sous le contrôle de BASIC.

Donnons également quelques indications sur *l'assembleur*, avant d'y revenir plus en détail dans la suite du livre.

Il s'agit d'un langage très proche du langage machine : les mots ne sont plus des octets mais des «mnémoniques» formés de trois lettres ou guère plus, rappelant l'instruction qu'ils symbolisent. Ainsi DECA est une instruction qui demande à l'unité de calcul de diminuer le contenu de l'accumulateur A d'une unité, de le *décrémenter.*

Il s'agit donc d'abord d'une copie du langage machine. Mais beaucoup d'assembleurs offrent un «plus» : par exemple, la possibilité d'utiliser des symboles pour les constantes qui seront utilisées dans un programme.

L'assembleur proprement dit est un programme (écrit en langage machine, le seul que connaisse vraiment le microprocesseur) qui décode une suite de mnémoniques et opère la traduction en langage machine. En définitive, un programme écrit avec un assembleur est plus long qu'un programme en langage machine, mais il est plus agréable à lire, et beaucoup plus facile à comprendre, à cause des mnémoniques.

Les langages de haut niveau

On nomme ainsi tous les langages suffisamment éloignés du langage machine : cela leur permet d'exprimer des instructions beaucoup plus complexes et subtiles que les instructions du microprocesseur. Et pour cela, ils utilisent des mots de tous les jours (hélas issus de l'anglais la plupart du temps).

Ces langages de programmation sont maintenant très nombreux avec de multiples dialectes. La plupart sont des langages spécialisés en vue d'applications industrielles, scientifiques ou commerciales.

Parmi les langages les plus connus et les plus utilisés, on peut citer le vénérable FORTRAN, défini en 1956, le COBOL, adapté aux problèmes de gestion. Les micro-ordinateurs utilisent plutôt des langages universels, comme le PASCAL ou le BASIC.

C'est ce dernier qui, à l'heure actuelle, est le plus connu : son succès vient notamment de sa simplicité, mais on lui reproche de donner de mauvaises habitudes de programmation. Il y a aussi des querelles linguistiques dans le monde des informaticiens.

Revenons maintenant au problème de la traduction ; un programme est écrit dans un langage de haut niveau, BASIC par exemple : comment se fait le passage au langage machine ? Ce n'est pas aussi simple que pour le langage assembleur : il y a alors une correspondance presque terme à terme entre les mnémoniques et les instructions du langage machine. Une instruction BASIC se traduit souvent en plusieurs dizaines d'instructions du microprocesseur. En gros, on trouve deux modes de traduction : la *compilation* et l'*interprétation*.

Le premier mode de traduction semble plus naturel ; un programme spécial, appelé compilateur, opère la traduction de tout un programme BASIC en un programme en langage machine : ce programme est alors mis en mémoire, et, lorsqu'on désire une exécution, c'est le programme en langage machine qui est mis en oeuvre. Le programme BASIC ne sert plus.

Le second mode de traduction est très différent : le programme BASIC est mis en mémoire, sans être traduit. Mais chaque fois que l'on veut l'exécuter, chaque instruction BASIC est d'abord traduite, puis exécutée. La traduction et l'exécution sont intimement liées.

La conséquence la plus claire est que les programmes BASIC sont alors exécutés plus lentement, il faut tenir compte du temps de traduction. Mais les BASIC interprétés sont aussi plus souples d'emploi : en cas d'erreur, on signale à l'utilisateur l'endroit précis de l'erreur et sa signification. Pour un programme compilé, toute erreur est fatale, l'ordinateur se « plante ».

Les BASIC des micro-ordinateurs familiaux sont tous interprétés, celui d'Alice en particulier. La rapidité d'exécution n'est pas l'essentiel quand on débute, et il est par contre essentiel de voir signalées les erreurs.

LE BASIC D'ALICE

Nous allons maintenant décrire le BASIC d'Alice ; c'est un BASIC « Microsoft », c'est-à-dire un des dialectes les plus courants parmi les micro-ordinateurs. Mais il est spécialement adapté aux caractéristiques d'Alice : toutes les instructions qui concerneront le graphisme, les couleurs, le son, tiendront compte de ces spécificités : nous les détaillerons plus tard.

C'est aussi un BASIC « simple » (par opposition à un BASIC « étendu ») ; on peut le constater, par exemple, en regardant son encombrement mémoire : le BASIC et le moniteur d'Alice occupent environ la moitié des 16 Ko de ROM, ce qui est peu. On trouve couramment des BASIC de 16 Ko ou plus.

Faisons un rapide résumé de ses possibilités non banales :

— il contient des ordres graphiques (en basse résolution, 32 x 64 ou en haute résolution, 160 x 125) ;

— il contient une instruction musicale SOUND ;

— il permet des branchements multiples (ON...GOTO et ON...GOSUB) et accepte plusieurs instructions par ligne ;

— il contient des opérateurs logiques AND, OR, NOT et offre la possibilité de lire et d'écrire en mémoire (PEEK et POKE).

Par contre, on regrettera principalement :

— l'absence de ELSE, après IF...THEN ;

— les scientifiques déploreront l'absence de double précision et de certaines fonctions mathématiques (comme ATN). Mais soyons honnêtes, il n'est pas dans la vocation d'Alice de faire des calculs compliqués.

LE CLAVIER
L'EDITEUR BASIC

UN CLAVIER COMPLIQUE MAIS PRATIQUE

Comme c'est le cas pour beaucoup d'ordinateurs d'initiation, le clavier d'Alice est d'un emploi assez complexe... Cela peut paraître paradoxal, mais l'est moins si l'on songe que les ordinateurs professionnels sont utilisés très souvent par des dactylos émérites qui exigent un clavier standard. Au contraire, les hobbyistes pratiquent souvent la frappe avec deux doigts : c'est pour eux que l'on a prévu de proposer les mots clés du BASIC directement au clavier.



Comme on le voit sur la photo, les touches sont disposées dans l'ordre « AZERTY », comme pour toutes les machines à écrire françaises. Mais en plus des lettres et chiffres, les touches contiennent divers mots ou symboles. Un exemple :



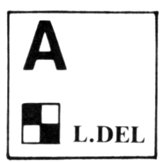
(Pour Alice-90. Dans le cas d'Alice version normale, le mot PRINT est écrit au-dessus de la touche).

- Une frappe normale :
[9] donnera sur l'écran 9

- Une frappe accompagnée de l'appui *simultané* de la touche [SHIFT] :
[SHIFT]-[9] donnera sur l'écran :)

- Une frappe accompagnée de l'appui *simultané* de la touche [CTRL] (abréviation de contrôle) :
[CTRL]-[9] donnera sur l'écran : PRINT

On voit donc que la capacité du clavier est multipliée par trois grâce à l'emploi de ces touches préfixes que sont [SHIFT] et [CTRL]. La touche [SHIFT] correspond au départ à la bascule majuscule/minuscule des machines à écrire ; ce n'est pas vraiment le cas pour Alice. Regardons la touche,



et essayons les trois frappes :

[A] donnera A

[SHIFT]-[A] donnera

[CTRL]-[A] ne donnera aucun caractère sur l'écran. Au contraire toute la ligne sera effacée : L.DEL signifie «line delete», effacement de la ligne.

L'affaire se complique si l'on considère que certains effets spéciaux peuvent être obtenus par des combinaisons de touches, et ne sont pas rappelés au clavier.

Ainsi [SHIFT]-[Ø] produit le passage en vidéo inversée pour les lettres : vert sur fond noir au lieu de noir sur fond vert. Un nouvel appui ramène au mode normal.

[CTRL]-[Ø] a pour effet de changer la couleur du curseur : au départ, le curseur est un rectangle noir clignotant sur fond vert, et qui indique la prochaine position de la lettre frappée. Après [CTRL]-[Ø], il clignote sur fond bleu, et si l'on poursuit la manoeuvre, on obtient les huit couleurs disponibles d'Alice ; attention, après un changement de couleur du curseur, les lettres resteront en noir et vert, mais les caractères semi-graphiques prendront la couleur du curseur.

[SHIFT]-[@] permet de suspendre temporairement l'exécution d'un programme ou un affichage sur l'écran. L'appui d'une autre touche annule son effet.

Les différents types d'écran

Ce que nous venons de décrire à propos du clavier n'est pas encore complet : ce n'est que le point commun entre les trois versions d'Alice. Avec Alice-90 et la nouvelle version d'Alice, on dispose d'autres possibilités, notamment des lettres minuscules.

Pour y accéder, il faut savoir d'abord qu'il y a quatre types d'écran :

— le mode 32 colonnes et 16 lignes, seul compatible avec la première version ;

— le mode 40 colonnes et 25 lignes : aucune différence, sinon que la fenêtre de travail s'est notablement agrandie ;

— le mode 80 colonnes et 25 lignes : la taille des lettres est d'autant réduite ;

— le mode « 81 » colonnes et 25 lignes : il n'y a en fait que 80 colonnes, mais à la différence du mode précédent, les lettres sont en vert sur fond noir.

Dans les deux derniers modes :

[SHIFT]-[Ø] ne donne plus les lettres en vidéo inversée, mais les *minuscules* ; malheureusement, sans les minuscules accentuées. Par ailleurs, les caractères semi-graphiques ont disparu et [CTRL]-[Ø] n'a plus d'effet. Nous verrons plus tard, dans le chapitre traitant du graphisme, que d'autres possibilités s'ouvrent quant aux changements de couleurs.

Pour passer d'un type d'écran à un autre, pas de difficulté : on tape l'instruction CLS (clear screen) qui efface l'écran, et on la fait suivre des nombres 32, 40, 80, ou 81 suivant le cas.

L'éditeur de ligne

Sur le clavier, on dispose de quatre touches de flèches, accessibles avec le prefixe [CTRL], ou bien, et seulement dans le cas d'Alice-90, accessibles directement ; on est vite déçu : seule la touche [←] semble avoir de l'effet, en déplaçant le curseur vers la gauche et en effaçant le dernier caractère écrit ; [CTRL]-[Z], ↑ apparaît sur l'écran mais c'est le symbole d'une opération (élévation à une puissance).

On ne peut donc promener le curseur sur l'écran et corriger ainsi des fautes de frappe.

Quand on écrit un programme BASIC il y a quand même, et heureusement, une possibilité d'activer ces touches : on dispose, contrairement à la première version d'un éditeur de ligne.

Sa mise en oeuvre est simple : [*] suivi du munéro de la ligne que l'on désire corriger, puis [ENTER], et l'on est en mode édition. Le curseur clignote, mais en laissant apparaître le caractère qui est en-dessous. On peut alors :

— déplacer le curseur avec [←] et [→] ;
— supprimer un caractère avec [↓] ;
— insérer un ou plusieurs caractères à la gauche du curseur : il suffit de les taper, le reste de la ligne sera automatiquement décalé ;
— on valide la ligne corrigée par [ENTER].

Signalons également que l'on peut provoquer la répétition automatique des touches par [SHIFT]-[ESPACE], l'arrêter par [CTRL]-[ESPACE], ce qui est bien pratique quand on doit déplacer rapidement le curseur.

Exemple d'utilisation : duplication de ligne.

Supposez que l'on doive recopier une ligne 2Ø, un peu longue en une ligne 25 ; voir par exemple le programme de musique un peu plus loin :

```
2Ø  DATA 16Ø,141,135,141,1Ø2,141,135,141
```

On tapera [*] [2] [Ø] puis [ENTER] ; on déplacera le curseur vers la droite, on tapera [5] et effacera le Ø : la ligne peut être validée. Bien sûr, la ligne 2Ø n'a pas disparu et reste dans le programme.

LES NOMBRES

Alice ne connaît qu'un seul type de nombres : les nombres au format «virgule flottante». Il n'y a pas de format entier ou double précision. La notation scientifique est utilisée pour les nombres plus grands que 999 999 999 ou pour les nombres plus petits que 0.01 :

E+07 signifie qu'il faut «pousser» la virgule de 7 rangs vers la droite :

$$3.1415E+7 = 31415000$$

E−04 signifie qu'il faut pousser la virgule de 4 rangs vers la gauche :

$$9.99E-04 = 0.000999$$

Alice accepte les écritures abrégées E7 à la place de E+07.

Les nombres sont affichés avec neuf chiffres significatifs, ce qui est largement suffisant pour les programmes courants. Il existe un «infini» pour Alice : le plus grand nombre qu'elle puisse envisager est 2^{127} (2 puissance 127), c'est-à-dire environ 10^{38} (un 1 suivi de 38 zéros...)

De la même façon, le plus petit nombre que connaisse Alice est 2^{-127}, c'est-à-dire environ 10^{-38}. Il y a une petite différence de comportement entre cet infiniment grand et cet infiniment petit : si on atteint la limite 2^{127}, Alice interrompt l'exécution du programme et signale une erreur de débordement (OV ERROR). Si ce résultat est trop petit, Alice l'assimile à 0 et poursuit tranquillement son programme. Dernière particularité, on peut parfaitement insérer des blancs dans l'écriture d'un nombre.

FONCTIONS DE TRAITEMENT DES NOMBRES

Opérations

+ addition
− soustraction
* multiplication
/ division
↑ puissance

ABS

Valeur absolue du nombre
Ex : ABS(– 48) = 48
ABS(51.5) = 51.5

INT

Partie entière du nombre (plus petit entier inférieur)
Ex : INT(3.25) = 3
INT(– 4.25) = – 5

SGN

Donne le signe d'un nombre
– 1 s'il est négatif
+ 1 s'il est positif
0 s'il est nul
Ex : SGN(– 8) = – 1
SGN(5 – 5) = 0

SQR

Donne la racine carrée éventuellement approchée
Ex : SQR(9) = 3
SQR(3) = 1.73205081

L'appel de cette fonction pour un nombre strictement négatif conduit à une erreur.

SIN, COS, TAN

Donnent respectivement le sinus, le cosinus, la tangente du nombre représentant un angle exprimé en radians.

LOG, EXP

Donnent respectivement le logarithme néperien et l'exponentielle du nombre.

LES CHAÎNES DE CARACTERES

Elles ont une longueur maximale de 255 caractères... mais pas automatiquement.

— Au départ, la longueur maximale d'une chaîne de caractères est de 100 caractères : il faut réserver davantage de place avec l'instruction CLEAR pour utiliser des chaînes plus grandes.

— Les chaînes entrées au clavier ont une longueur maximale de 127 caractères, quelle que soit la place réservée par CLEAR : cela provient de ce que la «mémoire tampon» de ligne, qui contient tous les caractères tapés avant qu'ils ne soient enregistrés, est limitée à 127 caractères.

LES FONCTIONS DE TRAITEMENT DE CHAINE

LEN

Elle donne la longueur de la chaîne, c'est-à-dire le nombre de ses caractères, espaces compris.

Ex : LEN("ALICE") = 5
LEN("BASIC ALICE") = 11

VAL

Donne la valeur du nombre écrit au début de la chaîne. Si la chaîne ne commence pas par un nombre, VAL est nul.

Ex : VAL("10") = 10
VAL("10 FRANCS") = 10
VAL("1E3") = 1 000
VAL("E3") = 0

MID$

Prend un morceau quelconque de la chaîne.
Ex : Si A$ = "ALICE"
Alors MID$(A$,2,3) = "LIC"
à partir de la deuxième lettre prendre 3 lettres

LEFT$

Prend la partie gauche d'une chaîne.

Ex : Si A$ = "ALICE"
Alors LEFT$(A$,3) = "ALI"

RIGHT$

Prend la partie droite d'une chaîne.

Ex : Si A$ = "ALICE"
Alors RIGHT$(A$,3) = "ICE"

+

Permet de mettre deux chaînes bout à bout (concaténation).

Ex : A$ = "BON" B$ = "JOUR"
A$ + B$ = "BONJOUR"

STR$

Pour mettre un nombre sous forme de chaîne

Ex : STR$(1984) = " 1984"
Attention : un espace devant le nombre

LES IDENTIFICATEURS

Ce sont les noms des variables. Il y a deux types d'identificateurs :

— Pour les variables numériques, n'importe quelle succession de lettres ou de chiffres constitue un identificateur, à condition qu'elle commence par une lettre.

Ex : X, ALICE, X1, DEBUT

— Pour les variables chaînes de caractères, même règle, mais le nom doit se terminer par $.

Ex : X$, NOM$, A1$

Ces deux règles sont communes pratiquement à tous les BASIC. Mais Alice impose des contraintes supplémentaires :

— L'interpréteur ne lit que les deux premiers caractères de chaque identificateur; pour lui, DEBUT et DERNIER représentent la même variable numérique.

— Il ne peut y avoir aucun mot du vocabulaire BASIC à l'intérieur d'un identificateur : si on utilise des variables s'appelant **TO**TAL, IN**FOR**MATION ou **SAND**ALE, on ne coupe pas d'une SN ERROR (erreur de syntaxe). Il faut faire attention aux mots clés cachés, ceux qui ne sont pas dans le manuel... Ce sont VARPTR et OFF.

Plus grave, l'écriture de l'affectation EXECUTION = 10 conduit Alice à la perplexité la plus totale, le curseur disparaît... Il ne reste plus qu'à réinitialiser (touche INIT).

Explication : Alice cherche à exécuter un programme en langage machine, programme qu'on ne lui a pas fourni.

Ce refus des mots clés est une contrainte. On risque de perdre beaucoup de temps à identifier les erreurs de syntaxe si l'on n'est pas prévenu. Il y a un avantage : l'interpréteur BASIC reconnaît les mots clés où qu'ils se trouvent, pas besoin de les encadrer de « blancs ». Cela permet d'écrire des programmes plus compacts, qui tiennent moins de place en mémoire.

On peut organiser les variables numériques ou chaînes en *tableaux*, ayant un nombre quelconque de dimensions, ce qui est tout à fait remarquable. Les mêmes règles s'appliquent aux identificateurs de tableaux, mais il faut leur réserver de la place par l'instruction DIM.

DIM A(5,2) réservera de la place pour 18 variables, puisque les indices iront de 0 à 5 et de 0 à 2.

LES COMMANDES

RUN démarre l'exécution d'un programme.
RUN 1ØØ : l'exécution commence à la ligne 1ØØ.

Attention, RUN réinitialise les variables à 0, ou à la chaîne vide. Si on veut l'éviter, il convient d'utiliser GOTO 1ØØ. On peut aussi provoquer une interruption par STOP.

CONT permet de continuer l'exécution après une interruption causée par l'appui de la touche [BREAK], ou par l'instruction STOP.

CSAVE, CSAVE* pour sauver sur cassette un programme ou un tableau.

Ex : CSAVE "PROG1"
CSAVE*T

LLOAD, LLOAD* pour charger un programme, un tableau à partir d'une cassette.

LIST, LLIST donne la liste du programme sur l'écran, sur l'imprimante. On peut spécifier un n° de début ou de fin :

Ex : LIST-1ØØ : toutes les lignes jusqu'à 1ØØ
LIST 9Ø-1ØØ : toutes les lignes entre 9Ø et 1ØØ.

Cet affichage est très rapide. On peut le suspendre au moyen de l'appui simultané de [SHIFT] et de [@], mais il faut être très rapide...

SKIPF identifie un programme écrit sur cassette, mais ne le charge pas :

Ex : SKIPF :LOAD permet de charger le second programme de la cassette.

NEW détruit le programme qui est en mémoire.

Ces commandes peuvent souvent être utilisées à l'intérieur d'un programme; cela peut donner des résultats surprenants.
Par exemple la ligne

```
1ØØ NEW
```

permet de faire un programme qui s'autodétruit lorsqu'il arrive à la ligne 1ØØ.

LES INSTRUCTIONS

CLEAR, suivie d'un entier, inférieur au nombre d'octets disponibles en mémoire utilisateur, réserve de la place pour les chaînes de caractères. A défaut, 100 octets sont réservés.

DATA permet d'introduire une liste de données qui seront lues par READ.

DIM réserve de la place pour les tableaux.

END. Instruction de fin de programme. Cette instruction est facultative.

FOR...TO...STEP...NEXT. Instructions de boucle; la panoplie est complète, on peut utiliser un pas négatif.

Attention, en cas d'impossibilité, la boucle est néanmoins exécutée une fois :

```
1Ø  FOR I=5 TO Ø
2Ø  PRINT"A"
3Ø  NEXT
RUN
```

à l'exécution :
A
OK

Comme indiqué dans cet exemple, le nom de la variable de contrôle est facultatif après NEXT.

GOSUB, GOTO. Branchement à un sous-programme, à un numéro de ligne.

IF...THEN. Branchement conditionnel. Pas de ELSE mais possibilité d'instructions multiples après THEN. La condition qui suit IF peut être complexe. On utilise alors les opérateurs AND, OR, NOT.

Ex : `IF A$="A" OR A$="E" THEN 1ØØ`
(le GOTO qui suit un THEN est facultatif)

INKEY$. «Variable» un peu spéciale : elle contient le caractère frappé au clavier; si aucun caractère n'a été frappé, elle contient la chaîne vide. Utile si on veut saisir un caractère sans utiliser INPUT.

Ex :
```
10  A$=INKEY$
20  IF A$="" THEN 10
```

Cela permet de «boucler» tant qu'une touche n'a pas été appuyée.

INPUT. Instruction d'entrée, permet d'initialiser une variable. Peut être assortie d'un «message».

LET, très démodée, c'est l'instruction d'affectation.

Ex : `LET PI=3.1415926`

Elle est facultative, et donc très peu utilisée.

ON...GOTO...(GOSUB). Instruction de branchement multiple. Si la variable qui suit ON ne prend pas une valeur acceptable, le programme ignore le branchement et se poursuit en séquence.

PRINT (TAB(n))(@), LPRINT. Instruction d'impression. Pour plus de détails sur PRINT @, voir le chapitre «Graphique». LPRINT concerne l'imprimante. Avec le séparateur « ;» l'écriture se fait à la suite, sans espace.

Avec le séparateur «,» l'écriture se fait en passant à la moitié suivante de l'écran. TAB(n) permet de positionner à la colonne n. Le nombre total de colonnes est de 32, 40 ou 80 suivant les cas. Voir chapitre «Graphique».

READ lit les données introduites par DATA.

REM, pour introduire un commentaire. Ne s'abrège pas en«'» (apostrophe) comme c'est parfois le cas.

RESTORE repositionne en début de DATA; la prochaine instruction READ lira la première donnée du programme.

RETURN. Retour d'un sous-programme.

STOP met un «point d'arrêt» dans un programme. L'exécution reprend par CONT. Utile pour la mise au point, car, entre temps, on peut demander l'impression des valeurs des variables.

MUSIQUE ET GRAPHISME



MUSIQUE

Comme tout micro-ordinateur familial qui se respecte, Alice permet de faire de la musique. L'instruction SOUND F,D donne l'ordre d'émettre un son de durée D, de «fréquence» F.

• La durée peut être un entier de 1 à 255 ; on peut chronométrer : lorsque D vaut 1, le son dure un peu moins de 7/100 de seconde. Il faudra donc prendre D = 14 pour avoir un son d'environ 1 seconde et lorsque D = 255, le son durera près de 17 secondes. Comme il y a proportionnalité entre D et la durée réelle, il suffit de décider d'une durée D1 pour une croche, la durée d'une noire sera 2 * D1, celle d'une blanche 4 * D1, etc.

• La fréquence F est également un entier entre 1 et 255. Le son est d'autant plus aigu que F est élevé, et la plage des sons s'étend sur plus de trois octaves. Malheureusement, le nombre F n'est pas proportionnel à la fréquence réelle du son. On sait en effet que les fréquences doublent quand on change d'octave, mais si on essaie :

```
SOUND 5Ø,1Ø
SOUND 1ØØ,1Ø
SOUND 2ØØ,1Ø
```

Pas besoin d'avoir l'oreille musicale pour constater qu'il ne s'agit pas de la même note sur trois octaves. Il faut donc se reporter au tableau de correspondance fourni par le manuel. Nous vous proposons ainsi la transcription d'un prélude de J.S. BACH ; gros avantage pour la programmation, la durée des notes est constante, ce qui permet de diviser le nombre de données par deux.

Remarquez l'utilisation du couple DATA-READ et bien sûr, de l'éditeur pour dupliquer les lignes.

```
1Ø   * PRELUDE J.S. BACH *
2Ø  DATA 18Ø,127,119,127,1Ø2,127,119,127
25  DATA 18Ø,127,119,127,1Ø2,127,119,127
3Ø  DATA 16Ø,141,135,141,1Ø2,141,135,141
```

```
35  DATA 16Ø,141,135,141,1Ø2,141,135,141
4Ø  DATA 175,141,127,141,119,141,119,141
45  DATA 175,141,127,141,119,141,119,141
5Ø  DATA 18Ø,154,141,154,127,154,141,154
55  DATA 18Ø,154,141,154,127,154,141,154
6Ø  DATA 193,16Ø,154,16Ø,127,16Ø,154,16Ø
65  DATA 193,16Ø,154,16Ø,127,16Ø,154,16Ø
7Ø  DATA 189,148,135,148,119,148,135,148
75  DATA 189,148,135,148,119,148,135,148
8Ø  DATA 189,154,148,154,119,154,148,154
85  DATA 189,154,148,154,119,154,148,154
9Ø  DATA 18Ø,135,119,135,1Ø2,135,119,135
95  DATA 18Ø,135,119,135,1Ø2,135,119,135
1ØØ  DATA 171,141,127,141,119,141,127,141
1Ø5  DATA 171,141,127,141,119,141,127,141
11Ø  DATA 171,154,141,154,127,154,141,154
115  DATA 171,154,141,154,127,154,141,154
12Ø  DATA 16Ø,154,141,154,127,154,141,154
125  DATA 16Ø,154,141,154,127,154,141,154
13Ø  DATA 16Ø,119,1Ø2,119,83,119,1Ø2,119
135  DATA 16Ø,119,1Ø2,119,83,119,1Ø2,119
14Ø  DATA 154,83,62,83,127,83,62,83
145  DATA 154,83,62,83,127,83,62,83
15Ø  DATA 141,1Ø2,83,1Ø2,72,1Ø2,83,1Ø2
155  DATA 141,1Ø2,83,1Ø2,72,1Ø2,83,1Ø2
16Ø  DATA 141,119,1Ø2,119,93,119,1Ø2,119
165  DATA 141,119,1Ø2,119,93,119,1Ø2,119
17Ø  DATA 141,119,1Ø2,119,93,119,1Ø2,119
175  DATA 141,119,1Ø2,119,93,119,1Ø2,119
18Ø  DATA 127,1Ø2,93,1Ø2,5Ø,1Ø2,93,1Ø2
185  DATA 127,1Ø2,93,1Ø2,5Ø,1Ø2,93,1Ø2
19Ø  DATA Ø
2ØØ  REM ** PROGRAMME **
21Ø  READ N
22Ø  IF N=Ø THEN RESTORE :GOTO 21Ø
23Ø  SOUND N,1Ø :GOTO 21Ø
```

Une dernière remarque, il est possible de manipuler le générateur de son d'Alice pour obtenir des sons ou effets intéressants, mais pour cela on doit passer par l'assembleur, question de rapidité d'exécution.

GRAPHISME

Dessiner avec Alice? On a le choix... Semi-graphisme, graphisme basse résolution, graphisme haute résolution. Suivant le type d'écran :

32 colonnes et 40 colonnes : semi-graphisme ou basse résolution; 80 colonnes et 81 colonnes : haute résolution.

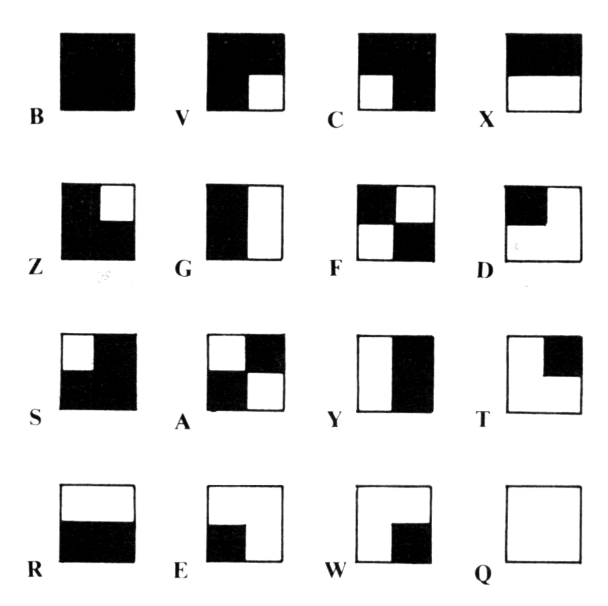
Le semi-graphisme

Il y a encore quelques années, les ordinateurs étaient entre les mains de gens trop sérieux pour s'amuser à dessiner, et rien n'était prévu à cette fin. On était obligé de se contenter d'images à base de lettres ou symboles soigneusement agencés.

Puis vinrent les micro-ordinateurs familiaux. Par exemple, un des tous premiers, le P.E.T. : il disposait en plus des lettres et symboles habituels, d'une grande variété de caractères semi-graphiques : formes géométriques variées, et même symboles des cartes à jouer.

Actuellement, on appelle semi-graphisme tout mode de dessin qui met en oeuvre l'instruction PRINT et ses variantes, et utilise les caractères habituels et des caractères *spéciaux* ou semi-graphiques.

Alice dispose de 16 caractères semi-graphiques, dans 8 couleurs de fond différentes :



Ils sont ici rangés dans un ordre « logique », qui correspond à l'écriture d'un nombre en base deux : à vous de le découvrir. Nous avons indiqué quelle touche du clavier permet de les obtenir (avec [SHIFT]).

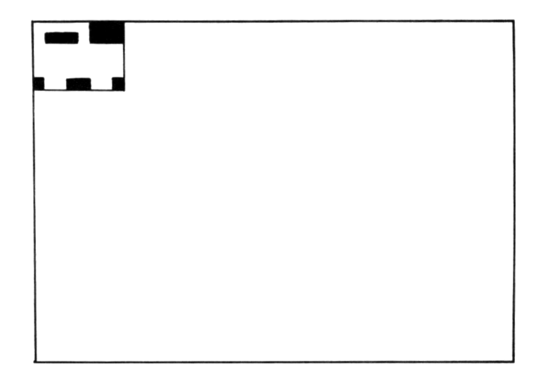
La combinaison de ces caractères permet de représenter des objets ou des personnages... mais très stylisés.

On utilise pour ces caractères spéciaux exactement la même syntaxe que pour les caractères ordinaires. Un premier exemple, en mode 32 ou 40 colonnes :

```
1Ø  CLS
2Ø  A$=" ▗▄▐█ "
3Ø  B$="      "
4Ø  C$=" ▖▗▖▗ "
5Ø  PRINT A$ :PRINT B$ :PRINT C$
```

Pour entrer les lignes 2Ø, 3Ø et 4Ø, on change la couleur du curseur par [CTRL]-[Ø].

Ce programme affiche une voiture de couleur dans un rectangle noir. Mais tout le reste de l'écran est vert.



Si l'on remplace la ligne 1Ø par CLSØ, le bas de l'écran sera noir, mais la fin des trois premières lignes restera verte ; il faut en effet accepter la contrainte suivante : tout caractère, chaîne de caractères, nombre est affiché sur fond vert, et chaque retour de ligne complète la ligne avec des rectangles verts. On peut éviter cet inconvénient en ne revenant pas à la ligne : avec point-virgule à la fin d'une instruction PRINT, et en utilisant l'instruction PRINT @, pour afficher nos chaînes de caractères n'importe où sur l'écran.

L'instruction PRINT @

PRINT et sa variante PRINT TAB fournissent des impressions *relatives* à la disposition courante du curseur. On peut éventuellement sauter des lignes (plusieurs instructions PRINT), se positionner à une colonne précise (PRINT TAB (colonne)), éviter le retour à la ligne (avec ;). Mais en aucun cas revenir en arrière sur une ligne, ou remonter plus haut sur l'écran.

Il existe aussi un mode d'affichage *absolu*. Toutes les cases de l'écran sont numérotées, et l'on peut alors préciser l'endroit X précis où doit commencer l'affichage, par PRINT @X,...

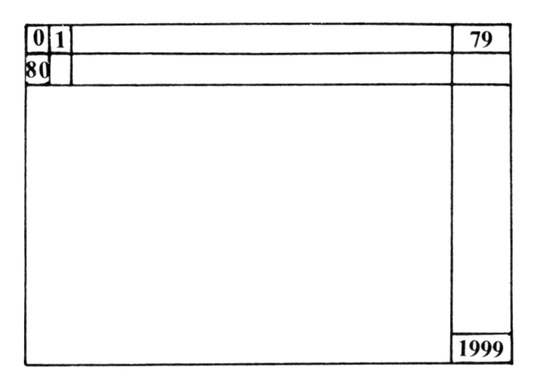
Voici, schématisé, le mode de numérotation :

0 1 2 31
32 33
511

mode 32

0 1 39
40
999

mode 40



mode 80 ou 81

Plus simplement, le numéro d'une case est donné par :
X = 32 * L + C où L et C sont les numéros de ligne et de colonne.
On commence par Ø; remplacer 32 par 40 ou 80 suivant les cas.

Donnons un exemple, en faisant se déplacer notre voiture :

```
10  CLSØ
20  A$=" ▄▄▄█ "
30  B$=" □□□□ "
40  C$=" ▄▄▄▄ "
50  D$=" "
60  FOR X=161 TO 195 :GOSUB 1ØØ :GOSUB 2ØØ :NEXT
70  CLSØ :GOTO 5Ø
100  PRINT @X,A$;:PRINT @X+4Ø,B$;:
     PRINT @X+8Ø,C$;:RETURN
200  PRINT @X−1,D$;:PRINT @X+ 39,D$;:
     PRINT @X+79,D$;:RETURN
```

La ligne 1ØØ est un sous-programme qui affiche la voiture, en imprimant ses trois éléments constitutifs sur trois lignes successives.

La ligne 2ØØ efface la «trace» laissée par l'image précédente, créant ainsi une impression de mouvement.

Semi-graphique et CHR$

Les caractères semi-graphiques peuvent être obtenus au clavier, au prix d'un certain nombre de manipulations quand on veut changer de couleur, mais on y arrive.

Il existe une autre possibilité. Chaque caractère, ordinaire ou spécial, porte un numéro de 32 à 255. On obtient le numéro d'un caractère par la fonction ASC (abréviation de A.S.C.I.I., le standard de cette numérotation). Mais réciproquement, on peut récupérer un caractère si on connaît son numéro, par CHR$.

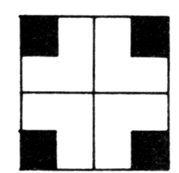
En essayant l'instruction PRINT CHR$(I), pour I de 32 à 255, on contemple toute la police des caractères disponibles. Dont certains d'ailleurs ne peuvent pas être obtenus au clavier, tels les [,] ou ←.

Les caractères semi-graphiques sont codés par les numéros de 128 à 255, par séries de 16 suivant la couleur du fond.

On peut utiliser cette propriété pour obtenir des changements de couleurs extrêmement rapides. Le programme suivant affiche au centre de l'écran une croix de couleur. Les couleurs changent tellement vite qu'il faut une boucle de ralentissement pour y voir quelque chose.

```
10  CLS0
20  X=500
30  FOR I=0 TO 112 STEP 16 :GOSUB 100
40  FOR J=1 TO 100 :NEXT J
50  NEXT I
60  GOTO 30
100  PRINT @X,CHR$(I+135);CHR$(I+139);
110  PRINT @X+40,CHR$ (I+141);CHR$(I+142);
120  RETURN
```

La croix est formée de quatre caractères semi-graphiques :



Le graphisme basse résolution

Disons-le tout de suite, ce graphisme n'apporte pas grand chose de plus que le semi-graphisme du paragraphe précédent : les instructions changent, le principe n'est pas le même, mais les images obtenues sont très semblables. En mode graphique, deux instructions seulement : l'une qui *allume* un point d'une couleur donnée, SET ; l'autre qui *éteint* un point, c'est RESET.

Tout point de l'écran est repéré par deux coordonnées, comme en mathématiques. Rappelons que le graphisme basse résolution ne concerne que les modes 32 ou 40 colonnes; il y a donc deux cas :

mode 32

0 20 63

0

10

(20,10)

31

mode 40

0 40 79

0

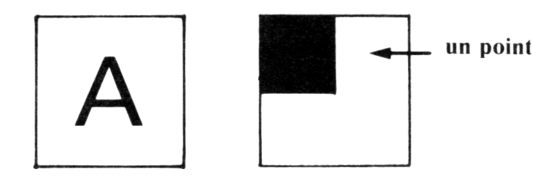
25

(40,25)

49

La coordonnée horizontale s'appelle l'abscisse, on a l'habitude de la noter X; la coordonnée verticale est l'ordonnée, notée Y.

Première remarque : il y a quatre fois plus de points graphiques que de positions possibles pour les caractères, donc un «point» a la même taille que le quart d'un caractère : c'est assez gros.



Le graphisme basse résolution doit toujours se faire sur un fond noir : CLSØ ; on dispose alors d'une palette de huit couleurs, du vert (n° 1) à l'orange (n° 8). Le point de coordonnées X,Y et de couleur C est allumé par :

SET(X,Y,C)

Il est éteint par :

RESET(X,Y)

Une ligne, horizontale par exemple, ne peut être tracée que point par point ; il n'existe pas d'instruction LINE, comme dans de nombreux BASIC.

En mode 40 colonnes :

```
FOR X=Ø TO 39 :SET(X,12,4) :NEXT X
```

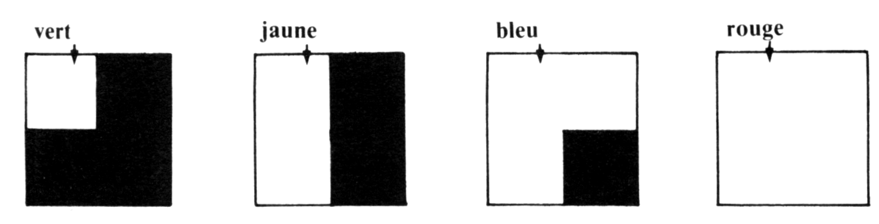
tracera une horizontale au milieu de l'écran, de couleur rouge.

En plus de la taille des points, une autre limitation de ce graphisme basse résolution est la *rigidité* des couleurs. Qu'entend-on par là ?

Un *pavé* de 4 points contigus ne peut faire apparaître que deux couleurs dont le noir ; ainsi le programme :

```
5  CLSØ
1Ø  SET(Ø,Ø,1)
2Ø  SET(Ø,1,2)
3Ø  SET(1,Ø,3)
4Ø  SET(1,1,4)
```

donnera successivement, en haut à gauche de l'écran :



Ainsi donc les points déjà allumés dans un pavé prennent la couleur du nouveau point que l'on allume.

C'est pourquoi d'ailleurs on ne fait du semi-graphique que sur fond noir : sur fond de couleur, SET n'allume que des pavés de quatre points contigus.

A quoi est due cette limitation?
Si on y réfléchit, et notamment avec l'exemple précédent, l'explication est claire : dans le mode graphique basse résolution, les instructions SET et RESET ont pour effet d'afficher un des 128 caractères semi-graphiques; comme ces caractères sont toujours noirs sur fond d'une autre couleur, la *rigidité* des pavés est expliquée.

Malgré ces défauts, le graphisme basse résolution offre des avantages : il est facile à programmer, et de plus, très rapide. Utilisé conjointement avec le semi-graphisme, il permet de programmer sans difficultés majeures des programmes de jeux.

Le graphisme haute résolution

Tout micro-ordinateur familial qui se respecte permet maintenant de faire du graphisme haute résolution. C'est le cas pour Alice avec 160 x 125 = 20 000 points adressables sur l'écran, ce qui commence à être conséquent.

Pour des raisons d'encombrement mémoire, et comme dans le graphisme basse résolution, il y a une certaine limitation dans la diversité des couleurs. Mais commençons par décrire ce qui se passe dans le mode 81 colonnes, assurément le plus simple.

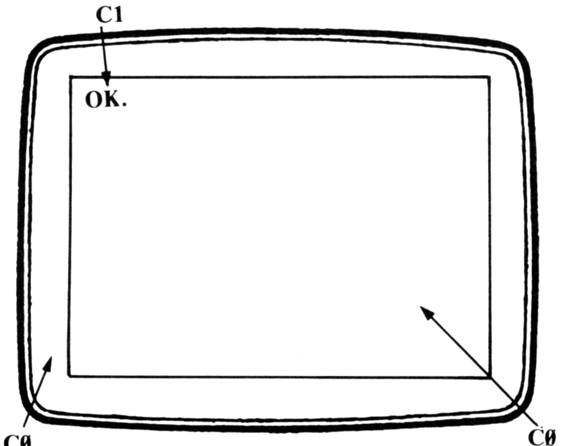
• Choix des couleurs : on ne peut disposer simultanément que de trois couleurs choisies parmi les huit, depuis le noir (n° 0) jusqu'au mauve (n° 7). L'orange, de numéro 8, n'est plus disponible. L'instruction permettant de choisir ces trois couleurs est :

SET * CØ,C1,C2, où CØ, C1 et C2 sont des entiers entre 0 et 7.

CØ s'appelle la couleur de *marge*, c'est en effet la couleur du bord de l'écran.

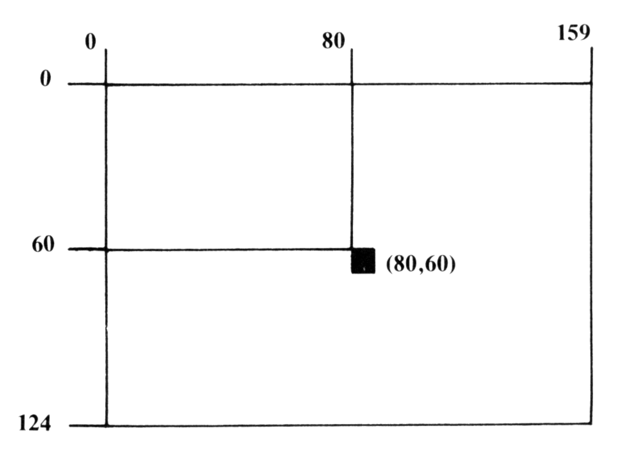
C1 s'appelle la couleur d'*intensité*, et C2 de *demi-intensité*.

En mode 81, le fond de l'écran prend toujours la couleur de marge, et est donc indiscernable du bord. Les caractères sont écrits dans la couleur d'intensité :



Attention donc, l'instruction SET ✻ ∅,∅,5 par exemple, produit un écran entièrement noir; c'est reposant pour les yeux mais on doit taper à l'aveuglette... Seul moyen de s'en sortir, une nouvelle instruction SET ✻.

• Les instructions graphiques : la syntaxe est semblable au cas de la basse résolution.

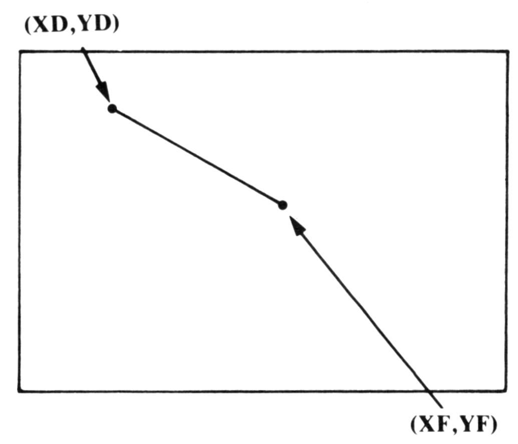


Un point est répéré par un couple (X,Y) de coordonnées comme indiqué sur le schéma. L'instruction SET(X,Y,C) demande cette fois que C soit un des entiers 0, 1 ou 2. Elle allume un point de couleur C∅ si C=∅, C1 si C=1, C2 si C=2.

Ces couleurs dépendent donc de la dernière instruction SET , et toute nouvelle instruction SET * changera la couleur des points déjà allumés. Bien sûr, comme le fond de l'écran est de couleur C∅, les points allumés ne seront vus que s'ils sont de couleur C1 ou C2. Ainsi SET(1∅∅,1∅∅,∅) a le même effet que l'instruction d'effacement : RESET(1∅∅,1∅∅).

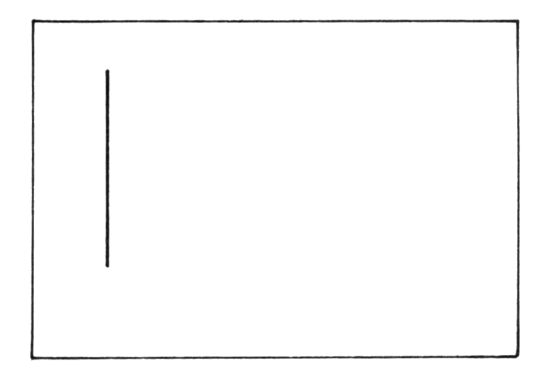
• Des lignes : maintenant que nous disposons d'une meilleure résolution, regardons quelques exemples de tracés; et pour commencer, de tracés de lignes.

Le problème est le suivant : en n'utilisant que l'instruction SET, comment joindre un point de départ (XD,YD) et un point d'arrivée (XF,YF)?



Pas de problème si le segment est horizontal ou vertical. Nous avons déjà traité le cas en basse résolution; il suffit d'allumer tous les points de la ligne ou de la colonne. Exemple :

```
FOR Y=5 TO 1∅∅ :SET(2∅,Y,1) :NEXT Y
```



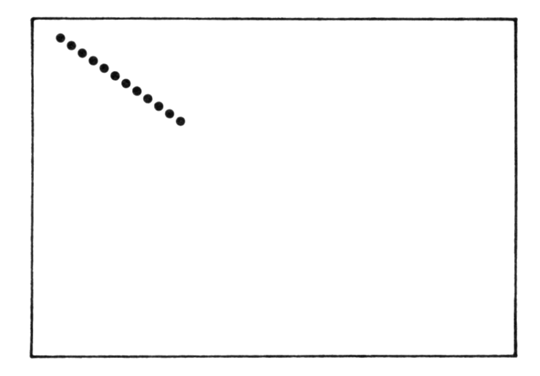
Mais lorsqu'il s'agit d'une oblique, le problème se complique. Première idée, on allume un point par colonne en utilisant une boucle :

```
FOR X=XD TO XF :SET(X,  ,1) :NEXT X
```

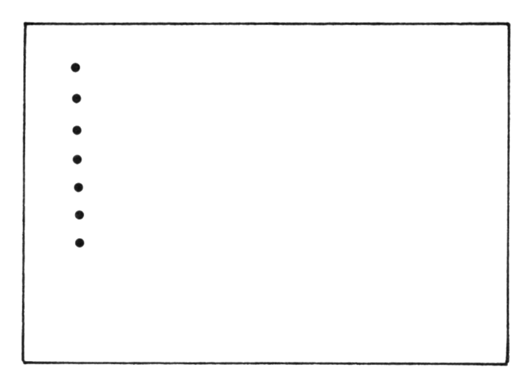
Les lecteurs ayant un peu de souvenirs de leurs cours de maths se rappelleront peut-être que cela revient à considérer Y comme fonction de X : Y = A * X + B, et l'on calcule A et B avec les points de départ et d'arrivée. Tout cela peut faire un sous-programme :

```
1ØØ  REM ** TRACE D'UNE LIGNE **
11Ø  A=(YF-YD)/(XF-XD) :B=YD-A * XD
12Ø  FOR X=XD TO XF :SET(X,A * X+B,1) :NEXT X
13Ø RETURN
```

Il faut bien sûr que XD soit plus petit que XF. Effet sur l'écran :



Mais cette méthode pose un problème : elle ne convient pas lorsque le segment est vertical (XD=XF); voir la division par 0 ligne 11Ø, elle est également déficiente quand il est proche de la verticale :

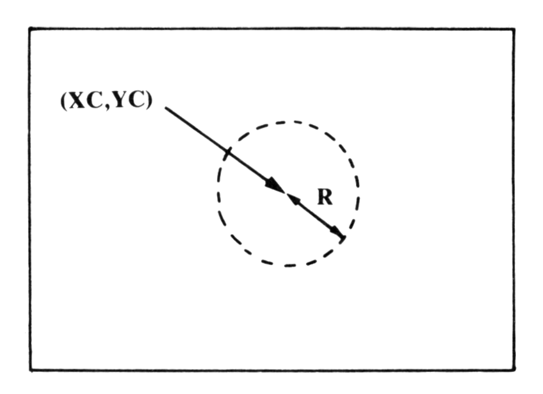


En effet, on n'allume qu'un point par colonne. On pourrait, bien sûr, échanger le rôle de X et de Y. Mais alors se poserait le problème des lignes horizontales... Proposons un autre «algorithme» qui fait intervenir le nombre de points que l'on désire allumer pour aller de (XD,YD) à (XF,YF); ce nombre doit être mis dans la variable NB avant l'appel du sous-programme.

```
100  REM ** TRACE D'UNE LIGNE **
110 FOR T=0 TO 1 STEP 1/NB
120  SET((XF-XD) * T+XD,(YF-YD) * T+YD,1)
130  NEXT T :RETURN
```

Pour les matheux, il s'agit d'une représentation paramétrique de la droite. Le résultat sur l'écran dépendra bien sûr de la valeur choisie pour NB, mais on pourra toujours avoir un tracé sans trou.

- Des cercles ou des ellipses : le problème est cette fois le tracé d'un cercle de centre (XC,YC) et de rayon R.



Il faut bien sûr s'arranger pour ne pas sortir de l'écran. Nous utiliserons encore une représentation paramétrique.

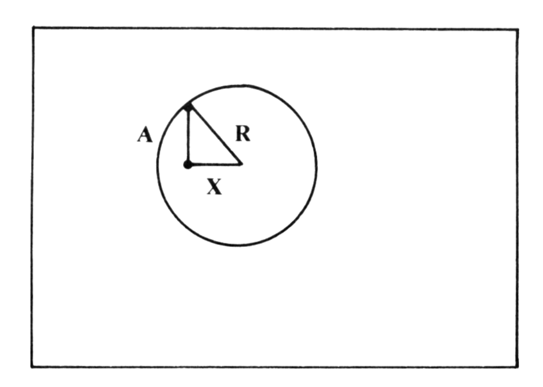
```
100  REM ** TRACE D'UN CERCLE **
110  FOR T=0 TO 6.28 STEP 6.28/NB
120  SET(XC+R * COS(T),YC+R * SIN(T),1)
130  NEXT
```

Pour un grand cercle (R = 40 par exemple), il faut plus de deux cents points pour un tracé presque continu... Et malgré tout, le résultat n'est pas sans irrégularités, le contour n'est pas parfaitement lisse. On ne peut guère éviter ce phénomène, sauf à augmenter considérablement la résolution.

Il est possible de déformer le cercle précédent en ellipse : on donne deux rayons R1 et R2, et on réécrit la ligne 12Ø :

```
12Ø  SET(XC+R1 * COS(T),YC+R2 * SIN(T),1)
```

Un cercle plein se dessine ligne par ligne, ou colonne par colonne. On utilisera alors le théorème de Pythagore :



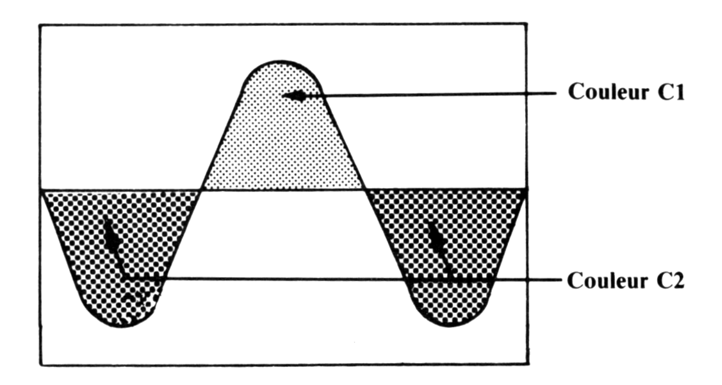
pour obtenir le programme suivant :

```
1ØØ  FOR X=-R TO R
11Ø  A=INT(SQR(R * R - X * X))
12Ø  FOR Y=-A TO A
13Ø  SET(XC+X,YC+Y,1) :NEXT Y :NEXT X
14Ø  RETURN
```

Suivant ce principe, toutes sortes de courbes pourront être tracées; un principe : on obtiendra des dessins plus agréables s'ils sont formés de segments verticaux ou horizontaux. Ainsi pour tracer P arches de sinusoïdes :

```
1ØØ  FOR X=Ø TO 159
12Ø  S=5Ø * SIN(3.14 * X * P/159)
125  IF S<Ø THEN 15Ø
13Ø  FOR Y=6Ø TO 6Ø+S
14Ø  SET(X,Y,1) :NEXT Y :GOTO 17Ø
15Ø  FOR Y=6Ø TO 6Ø+S STEP -1
16Ø  SET(X,Y,2) :NEXT Y
17Ø  NEXT X
```

sur l'écran :



Le mode 80 et les limites du graphisme

Explorons maintenant le mode 80; et d'abord le choix des couleurs. A l'appel CLS8Ø, on obtient un écran vert, des lettres noires et un bord noir. Si l'on essaie SET * CØ,C1,C2 avec des valeurs différentes, on constate que :

- CØ est toujours la couleur de marge, c'est-à-dire du bord de l'écran. C'est aussi la couleur du caractère.
- C1, couleur d'intensité, est aussi la couleur du fond de l'écran.
- C2 reste la couleur de demi-intensité, sans effet immédiat sur l'écran.

En résumé :

	CØ marge	C1 intensité	C2 1/2 intensité
mode 81	fond	caractère	

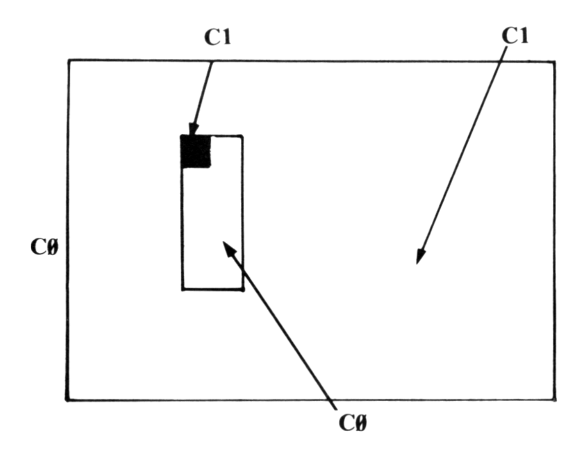
	CØ marge	C1 intensité	C2 1/2 intensité
mode 80	caractère	fond	

Regardons maintenant l'effet de :

SET(X,Y,1)

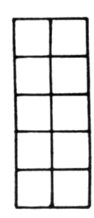
— en mode 81, cette instruction allumait un point de couleur C1 sur le fond de l'écran, qui était de couleur CØ.

— en mode 80, elle allume un point de couleur C1, dans un *rectangle* de couleur CØ.



L'instruction SET(X,Y,2) produira le même effet, mais en allumant un point de couleur C2 dans un rectangle de couleur CØ.

Le rectangle qui s'est affiché en mode 80 est en quelque sorte la mesure de la rigidité des couleurs; il a une taille de deux points de largeur sur cinq de hauteur :



et dans ce rectangle, ne peuvent coexister que deux couleurs : celle de la marge CØ, et une des couleurs C1 ou C2.

Bref, il y a une certaine limitation dans le choix des couleurs, comme pour le graphisme basse résolution. Mais cette limitation est moins apparente car les rectangles de rigidité sont relativement petits.

Et la différence entre le mode 80 et le mode 81 ?... Elle apparaît uniquement concerner la couleur des caractères et celle du fond de l'écran, de sorte que le mode 80 est une « vidéo-inversée » du mode 81 ; en tout cas, le graphisme haute résolution n'a d'intérêt qu'en mode 81.

PROGRAMMATION EN ASSEMBLEUR



L'EDITEUR

UN EDITEUR, POUR QUOI FAIRE ?

Nous parlerons bien sûr ici d'un programme éditeur ; même un peu limité, comme dans le cas d'Alice, un tel programme a beaucoup d'intérêt. Il permet de faire du « traitement de texte », c'est-à-dire pour simplifier, du courrier informatisé. Il fait de l'ordinateur une super machine à écrire. Mais son utilisation essentielle est de permettre l'écriture d'un programme en assembleur : c'est ce que nous aborderons dans le chapitre suivant.

Mise en route de l'éditeur

Le programme éditeur et le langage BASIC sont étanches l'un par rapport à l'autre. On ne peut simultanément utiliser des instructions BASIC et les commandes de l'éditeur. La bascule entre les deux par contre est aisée :

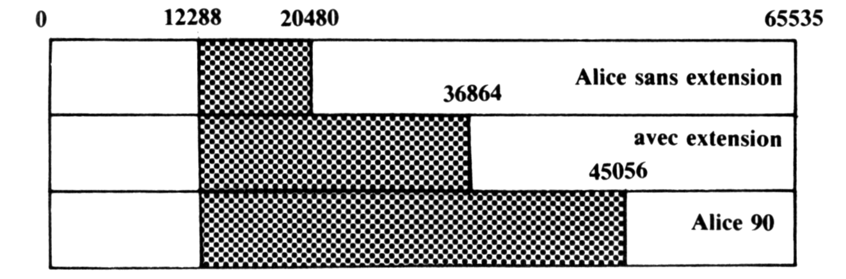
A partir du BASIC : [SHIFT]-[&] (suivi de [ENTER]) donne la main à l'éditeur.

A partir de l'éditeur : [BREAK] [BREAK] (deux appuis) redonne la main au BASIC.

Mais tout n'est pas si simple ; avant d'utiliser l'éditeur une première fois, il faut lui réserver de la mémoire vive, au moyen de l'instruction CLEAR.

Les cloisons mobiles

Le BASIC et l'éditeur ne doivent pas en effet utiliser les mêmes emplacements en mémoire vive. Or si l'on ne précise rien, le BASIC utilisera toute la mémoire vive disponible ; sachant que les adresses de toutes les cases mémoire disponibles vont de Ø à 65 535, on aura les schémas suivants :



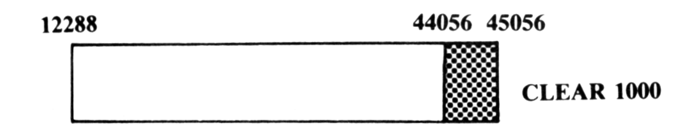
Les cloisons qui encadrent la zone disponible sont fixes. L'extérieur correspond à des adresses de mémoire morte (le langage BASIC, l'assembleur) ou bien à des petites zones de mémoire vive où le BASIC entrepose ses variables internes : sans information précise, il vaut mieux ne pas y toucher.

Lorsqu'un programme BASIC est écrit, il est entreposé dans la mémoire vive à des adresses basses : à partir de 13 1ØØ environ. Mais le BASIC utilise aussi le haut de la mémoire vive pour y stocker les chaînes de caractères, ce qui demande parfois beaucoup de place.

A l'initialisation, Alice réserve 100 octets pour ces chaînes, par exemple entre 44 955 et 45 Ø56 pour Alice-90. Il est possible d'étendre cette zone à l'aide de l'instruction CLEAR :

CLEAR 1ØØØ réservera 900 octets de plus.

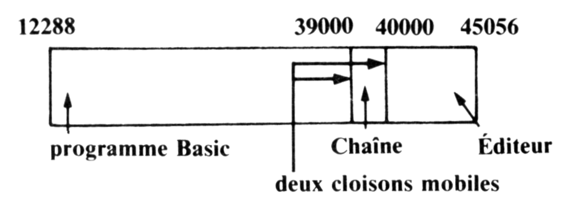
On peut le vérifier en interrogeant Alice sur sa mémoire disponible (instruction MEM) 31 814 octets avant, 30 914 octets après l'instruction CLEAR 1ØØØ. Si la place réservée aux chaînes est insuffisante, on obtiendra le message d'erreur OS ERROR (Out of string).



Si par contre le programme BASIC est trop important et vient à déborder sur la zone réservée aux chaînes, on aura le message OM ERROR (Out of memory).

Pour pouvoir utiliser l'éditeur, il faudra réserver de la place en mémoire vive, et cela se fera plus haut que la zone réservée aux chaînes. On le fera en donnant un second argument à CLEAR, l'adresse du début de cette zone :

CLEAR 1ØØØ,4ØØØØ donnera :



Il faut bien sûr adapter ces chiffres à la configuration dont on dispose. Nos exemples seront donnés pour Alice-90, avec toujours l'initialisation CLEAR1ØØØ,4ØØØØ, mais nous n'utiliserons qu'une faible partie des 5000 octets réservés à l'éditeur.

Les commandes de l'éditeur

Reprenons les choses au début. Après CLEAR 1ØØ,4ØØØØ, nous entrons dans l'éditeur par [SHIFT]-[&] puis [ENTER]. L'écran change de couleur : fond bleu foncé, caractères blancs. Et l'on peut commencer à écrire... On dispose de presque tous l'écran. Seule la dernière ligne est réservée au dialogue avec l'éditeur.

Tout se passe apparemment comme en BASIC, mais avec une facilité de correction très différente. Tout d'abord, les quatre touches de flèche fonctionnent !

[←] et [→] pour se déplacer sur la ligne, et ce sans effacer les caractères déjà écrits.

[↑] et [↓] pour se placer au début de la ligne précédente ou de la ligne suivante ; on s'est rapproché d'un éditeur pleine page idéal.

Voyons maintenant les autres commandes :

— pour insérer des caractères, commande CTRL-J, l'insertion se fait à gauche du curseur.

— pour insérer une ligne, commande CTRL-I, quand on est en début de ligne.

— pour effacer le caractère qui est sous le curseur, commande CTRL-D.

— pour effacer une ligne, commande CTRL-A (comme en BASIC).

Enfin, et cela est bien pratique notamment pour les flèches : CTRL-R permet de se mettre en mode répétition automatique. On revient au mode normal par la même commande.

Toutes ces commandes sont donc un peu différentes de celles de l'éditeur-BASIC. Il faut s'y habituer, mais la grande différence est que l'on travaille sur la totalité de l'écran et non sur une seule ligne de programme.

Quant aux types d'écran, ils se limitent ici à deux :

soit 40 colonnes (et donc des caractères semi-graphiques),

soit 80 colonnes (et donc des minuscules).

On passe de l'un à l'autre par CTRL-8.

Un exemple d'utilisation

Nous allons maintenant faire connaissance avec les dernières commandes de l'éditeur, qui n'ont pas d'équivalent en BASIC.

Mettons nous dans la peau d'un P.D.G., d'une petite entreprise.

A la suite de diverses péripéties, un (grand) nombre de clients envoient des lettres de réclamation : il faut leur répondre de façon personnalisée... mais sans y passer trop de temps.

Nous commençons donc par éditer une «lettre-type» :

Paris, le 7 Novembre 1984 Cher Monsieur * * * *, Nous avons bien reçu votre lettre du * * * *. Et nous sommes profondément désolé des ennuis que vous avez eus avec nos produits. Notre responsable de fabrication s'occupe activement de ce petit problème, et nous pensons pouvoir très bientôt vous faire savoir qu'il a été résolu. Acceptez encore, cher Monsieur * * * *, toutes nos excuses et veuillez agréer l'expression de toute notre considération.

On peut bien sûr en rajouter dans les formules de politesse, par exemple; l'essentiel est de faire une lettre «passe-partout».

Il s'agit maintenant de remplir les blancs : pour chaque client, on fera les manoeuvres suivantes :

[CTRL]-[L] (recherche de ligne) puis 1 : l'éditeur va rechercher la première ligne du texte et y positionner le curseur;

[CTRL]-[F] (recherche de mot) puis * * * * : l'éditeur recherche la première occurence de cette chaîne et y place le curseur; on peut alors remplacer nos étoiles par le nom du client.

En répétant la manoeuvre [CTRL]-[F], on pourra «personnaliser» entièrement la lettre.

Il ne reste plus qu'à l'imprimer :

[CTRL]-[6] permettra l'impression de toute la lettre; on peut utiliser [CTRL]-[H] pour une copie d'écran (hard copy).

Enfin, si vraiment ces problèmes «techniques» de notre petite entreprise ont tendance à se reproduire souvent... il faut enregistrer la lettre-type sur cassette par CTRL-3. L'éditeur demande de fournir un nom au fichier et, une fois le magnétophone prêt, on lance la sauvegarde par ENTER.

Bref, des possibilités nouvelles par rapport au BASIC; et même si l'on est pas P.D.G., que l'on pense au problème des cartes de voeux...

Les petits secrets de l'éditeur

Un des inconvénients de l'éditeur est son étanchéité avec le BASIC : il arrive que l'on veuille venir au BASIC, sans pour autant perdre le texte que l'on a écrit sous l'éditeur... On rencontrera souvent ce problème quand on écrira des programmes en assembleur : en effet, si l'on revient au BASIC par BREAK BREAK, puis si on retourne à l'éditeur par SHIFT-&, on se retrouve avec une page vierge... tout a disparu.

Il est possible de revenir à l'éditeur d'une autre façon : SHIFT-% puis ENTER.

En ce cas, le texte précédemment écrit est entièrement retrouvé.

On dit que cette entrée dans l'éditeur est une «entrée à chaud». Elle est *presque* toujours possible, mais il y a des exceptions.

Encore plus intéressant peut-être : les textes écrits sous l'éditeur ne sont pas directement utilisables par le BASIC, ce qui est dommage... On peut cependant les récupérer d'une certaine façon. Regardons comment est représenté en mémoire un tel texte. On peut pour cela utiliser un petit programme de vidage de la mémoire (ou de DUMP) :

```
1Ø  INPUT"ADRESSE DE DEPART :";A
2Ø  B=PEEK(A)
3Ø  PRINT A,B;CHR$(B)
4Ø  IF INKEY$="" THEN 4Ø
5Ø  A=A+1 :GOTO 2Ø
```

Dans ce petit programme, la ligne importante est la ligne 2Ø. PEEK(A) est le *contenu* de la case mémoire d'adresse A. En 3Ø sont affichés l'adresse, le contenu et s'il existe, le caractère de code B.

Maintenant, appelons l'éditeur, par CLEAR 1ØØ,4ØØØØ par exemple, et écrivons un texte. Puis retournons au BASIC et étudions la mémoire vive à partir de l'adresse 4ØØØØ ; on trouvera :

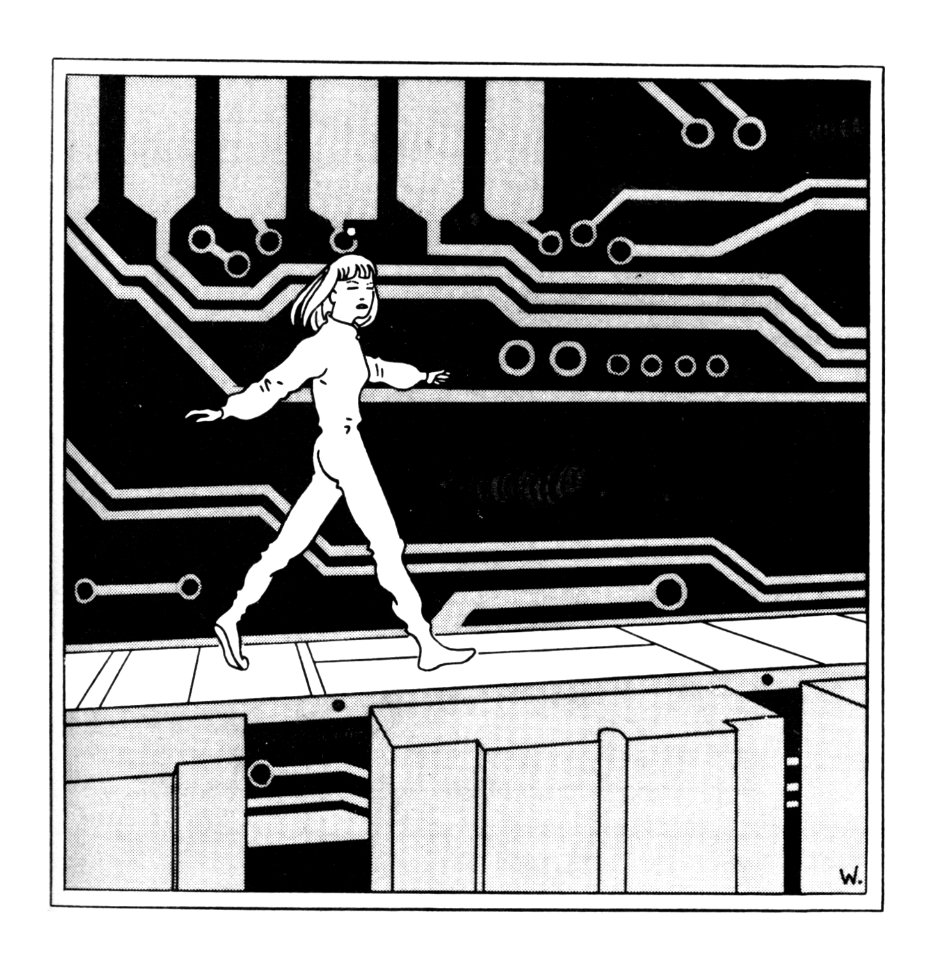
40000	7 ←		nombre de caractères de la ligne
40001	66	B	
40002	79	O	
40003	78	N	
40004	74	J	
40005	79	O	
40006	85	U	
40007	82	R	
40008	0 ←		
40009	9 ←		
.....			
40100	255 ←		fin du texte

D'où le programme suivant, qui permet de récupérer notre texte :

```
1Ø  I=4ØØØØ
2Ø  A=PEEK(I)
3Ø  IF A=255 THEN END
4Ø  FOR J=I+1 TO I+A :
5Ø  PRINT CHR$(PEEK(J));
6Ø  NEXT J :PRINT
7Ø  I=I+A+1 :GOTO 2Ø
```

Ce programme peut être modifié, pour qu'au lieu d'afficher chaque ligne, on la mette en mémoire.

L'ASSEMBLEUR



LE MICROPROCESSEUR 6803

Nous allons aborder maintenant la programmation en assembleur. Cela n'est guère plus difficile en fait que la programmation en BASIC, mais cela demande, au départ, un minimum de connaissances techniques. Il n'est pas nécessaire cependant d'aller jusqu'au niveau de l'électronique proprement dite, ou même de l'organisation des divers circuits imprimés, heureusement...

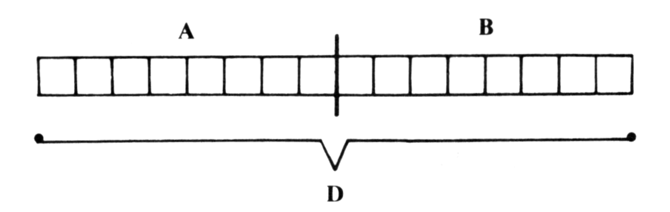
Commençons par décrire le microprocesseur 6803. Comme nous l'avons vu dans les premiers chapitres, le microprocesseur est un ordinateur à lui tout seul. Il contient :

• des mémoires, que l'on appelle plutôt *registres*. Elles portent un nom symbolique :

trois mémoires 8 bits :
A et B sont les accumulateurs
P est le registre d'état

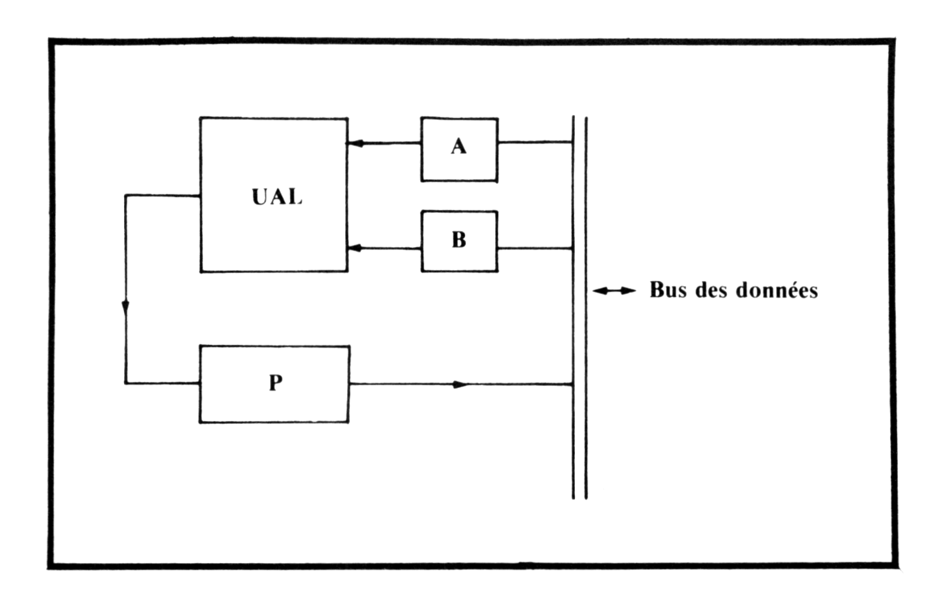
IX est le registre d'index
SP est le registre pointeur de pile (en anglais Stack)
PC est le registre compteur de programme (Program Counter)
Les trois derniers registres sont des mémoires 16 bits.

Il est parfois possible de considérer A et B comme deux moitiés d'un registre 16 bits :



Ce registre est alors nommé D; on dit que A est le poids fort, B le poids faible.

• Le microprocesseur contient aussi une *unité de calcul* : unité arithmétique ou logique, UAL. On peut, très schématiquement, montrer comment elle est reliée aux registres 8 bits :



Nous avons déjà parlé du bus des données; c'est par là que transitent les données, d'un octet, venant ou allant dans les mémoires, ou à l'intérieur du microprocesseur. Ce que nous avons voulu représenter sur cette figure ce ne sont pas les branchements réels mais les faits suivants :

— Tout calcul porte sur des opérandes dans A et B, ou l'un des deux seulement.

— Le résultat du calcul *influe* sur le registre d'état P; il retourne ensuite sur le bus des données, pour être stocké soit dans un registre, soit dans une case mémoire.

Pour résumer :
A et B sont les registres à tout faire ;
P est le « pense-bête » du microprocesseur ; il peut déceler si l'opération qui vient d'être faite a donné un résultat nul, a produit une retenue, etc.

• Le microprocesseur est dirigé par un cerveau, que l'on appellera logique de contrôle. Son rôle est fondamental.

— elle décode les instructions du programme,
— elle organise les transferts entre accumulateurs, entre les accumulateurs et la mémoire extérieure au microprocesseur ;
— elle lance l'exécution des instructions, des opérations.

En plus des registres et du bus des données, elle doit gérer le bus des adresses par lequel passent les adresses des mémoires. Toute cette sarabande est rythmée par les battements d'une horloge à quartz...

• Il nous reste maintenant à examiner le rôle des autres registres du microprocesseur.

Le registre IX est une mémoire que l'on utilise souvent pour traiter des adresses : nous savons, en effet que, pour un microprocesseur 8 bits, les données ont un format 8 bits (un octet) tandis que les adresses des cases mémoires RAM ou ROM sont codées sur 16 bits.

Le registre S est le pointeur de pile ; il contient l'adresse du sommet de la pile, c'est-à-dire de la zone de mémoire vive où le microprocesseur empile les résultats intermédiaires ou les données dont il risque d'avoir besoin.

Le registre PC : nous n'y toucherons pas directement ; il contient à tout instant l'adresse de la prochaine instruction à exécuter. C'est cette adresse que va lire la logique de contrôle quand elle vient de terminer l'exécution d'une instruction.

Un programme en langage machine

Après ces préliminaires, nous allons tout de suite montrer comment on écrit un programme en assembleur. Notre premier exemple sera très élémentaire : nous allons nous contenter de vérifier que le 6803 effectue correctement les opérations : addition, soustraction, multiplication.

Exposons notre plan :

Le programme va :

— lire des données dans les cases mémoires M1 et M2, et les charger dans A et B;
— effectuer l'opération demandée (ici l'addition);
— sauvegarder le résultat dans une mémoire R1.

Les instructions de chargement sont des abréviations du mot anglais LOAD; ainsi le début de notre programme sera :

LDAA M1 charger le contenu de M1 dans A
LDAB M2 charger le contenu de M2 dans B

Nous aurons ensuite l'instruction d'opération

ABA (addition de A et B)

On aurait pu utiliser SBA, pour soustraire B à A ou bien MUL, pour multiplier A et B;

Dans les deux premiers cas, le manuel de l'assembleur nous indique que le résultat est dans A, dans le dernier cas il est stocké en D : c'est normal, le produit de deux entiers codés sur un octet dépasse souvent 255 et doit être codé sur 16 bits.

Nous terminerons donc notre programme par

STAA R1 stocker le contenu de A dans R1

ou

STD R1 stocker le contenu de D dans R1

Le corps du programme est décrit, il manque l'environnement.

Le programme source

L'assembleur, rappelons-le, est un traducteur : il permet de traduire un programme écrit à l'aide des mnémoniques en un programme en langage machine.

Dans le cas d'Alice, il permettra aussi l'implantation dans la mémoire vive de ce programme.

On a l'habitude d'appeler :

— programme source : le programme écrit avec les mnémoniques,
— programme objet : le programme en langage machine.

Le programme source ne contient pas seulement la liste des instructions à exécuter; il faut en effet préciser à l'assembleur :

— *à quel endroit* il doit implanter le programme objet,
— *à quel endroit* se trouve la première instruction à exécuter.

On indique cela en ajoutant dans le programme source des *directives* : ce ne sont pas des instructions du microprocesseur, mais des consignes pour l'assembleur. Comment choisir cette adresse d'implantation? Rappelons ce que nous disions dans le chapitre sur l'éditeur : il faut éviter l'interpénétration des différents programmes.

Nous allons implanter notre programme objet dans la plage 4ØØØØ-45Ø56, mais en laissant le début de cette mémoire vive pour le programme source; choisissons par exemple l'adresse hexadécimale AØØØ, c'est-à-dire 4Ø96Ø; il faudrait modifier cette adresse si le programme source est un peu long.

Pourquoi hexadécimale? C'est le mode le plus adapté à l'écriture des programmes en assembleur, car il est très proche de la représentation binaire; en cas de difficulté, voir l'annexe... Tous les nombres intervenant en assembleur doivent être exprimés dans cette base, avec le préfixe $.

Ce choix d'adresse d'implantation se fait avec la directive

ORG $AØØØ (comme origine)

et cette directive est obligatoirement la première instruction du programme source.

Seconde directive : nous devons donner à l'assembleur l'adresse de la première instruction du programme objet à exécuter; plutôt que de donner directement cette adresse, nous allons lui attribuer une *étiquette*, c'est-à-dire un nom symbolique. Par exemple : DEBUT.

EXC DEBUT (comme exécution)

Nous devrons par la suite faire précéder la première instruction exécutable de cette étiquette, pour que l'assembleur s'y retrouve.

Occupons-nous maintenant des «variables» M1, M2 et R1 que nous avons librement utilisées ci-dessus. Il ne s'agit pas tout à fait des variables du BASIC : nous allons considérer que M1 et M2 symbolisent les adresses des cases mémoires où seront stockées ces données; il faut donc préciser à l'assembleur que l'on veut réserver ces cases mémoires; cela ce fait encore par une directive :

```
M1 DFO $Ø
M2 DFO $Ø
R1 DFO $Ø
```

Nous avons ainsi réservé trois cases mémoire, d'un octet chacune, et nous les avons en même temps initialisées à Ø . On peut également réserver une case mémoire double (c'est-à-dire deux octets consécutifs) par DFD. Continuons maintenant avec le corps du programme :

```
DEBUT      LDAA M1
           LDAB M2
           ABA
           STAA R1
           RTS
```

La dernière instruction RTS (retour de sous-programme) donnera au microprocesseur l'ordre de revenir au programme appelant : par exemple au programme BASIC.

Nous pouvons maintenant écrire ce programme avec l'éditeur.

Deux précautions :

— Les étiquettes doivent commencer en début de ligne et ne pas dépasser cinq caractères; les instructions et directives doivent commencer au moins au second caractère de la ligne.

— Il faut séparer les étiquettes, instructions, opérandes par au moins un blanc.

Nous éditerons ainsi notre programme :

```
        ORG     $AØØØ
        EXC     DEBUT
M1      DFO     $Ø
M2      DFO     $Ø
R1      DFO     $Ø
DEBUT   LDAA    M1
        LDAB    M2
        ABA
        STAA    R1
        RTS
```

zone étiquette	instruction ou directive	opérande

L'assemblage

Il faut maintenant enclancher le processus d'assemblage : c'est la commande CTRL-1.

Tout de suite un message :

LISTING (ECRAN, IMP, ENTER)?

Répondre par E puis ENTER pour avoir le résultat suivant sur l'écran :

```
ASSEMBLEUR ALICE REV 1.ØØ
COPYRIGHT MATRA, 1984


 1                                ORG     $AØØØ
 2                                EXC     DEBUT
 3  AØØØ   ØØ       M1            DFO     $Ø
 4  AØØ1   ØØ       M2            DFO     $Ø
 5  AØØ2   ØØ       R1            DFO     $Ø
 6  AØØ3   B6AØØØ   DEBUT         LDAA    M1
 7  AØØ6   F6AØØ1                 LDAB    M2
 8  AØØ9   1B                     ABA
 9  AØØA   B7AØØ2                 STAA    R1
1Ø  AØØD   39                     RTS
Ø  ERREUR(S)    PASSE1
Ø  ERREUR(S)    PASSE2
SYMBOLES :
M1=AØØØ     M2=AØØ1     R1=AØØ2
DEBUT=AØØ3
FICHIER OBJET ?
```

Détaillons un peu :

Première colonne, on trouve les numéros de ligne du programme source.
Seconde colonne, on trouve les adresses d'implantation des instructions du programme objet, de AØØØ à AØØD.
On trouve ensuite le code hexadécimal du programme objet; chaque instruction ou ligne du programme source est codée par un, deux ou trois octets.
Nous voyons que les directives ORG et EXC ne donnent lieu à aucun code objet : ce qui confirme que ce sont seulement des consignes à l'assembleur. Par contre les directives DFO ont un effet visible : l'initialisation des cases mémoire AØØØ,AØØ1 et AØØ2 à Ø.

Regardons la ligne 6 : B6 est le code de l'instruction LDAA, puis on trouve AØØØ, c'est bien l'adresse étiquetée M1 ; l'assembleur a joué son rôle.

Le reste du programme se lit sans problème. Que faire ensuite ? Au message FICHIER OBJET ?, on peut répondre en donnant un nom pour le programme objet : si le magnétophone est prêt, ce programme sera enregistré ; on ne pourra le récupérer qu'à partir du BASIC, par l'instruction CLOADM ; rappelons que le programme source peut lui aussi être sauvegardé, par une commande de l'éditeur.

On répond BREAK si l'on ne veut pas enregistrer. Nouvelle question : EXECUTION IMMEDIATE ? Si l'on répond oui, le programme est exécuté, et l'on retourne au BASIC. Sinon, retour à l'éditeur.

Rappelons qu'en cas de retour au BASIC, on peut retrouver le programme source en appelant l'éditeur par [SHIFT]-[%] et non [SHIFT]-[&].

La communication avec le BASIC

Notre programme a été exécuté, et, bien sûr, nous n'avons rien vu ; tel qu'il est écrit, il ne peut qu'additionner $Ø à $Ø, et stocker le résultat ; ce n'est encore qu'un squelette de programme.

Nous allons lui communiquer les opérandes et lire le résultat à l'aide d'un programme BASIC :

```
1Ø  ADR=4Ø96Ø
2Ø  INPUT "A=" ;A
3Ø  INPUT "B=" ;B
4Ø  POKE ADR,A :POKE ADR+1,B
5Ø  EXEC ADR+3
6Ø   PRINT PEEK(ADR+2)
7Ø  GOTO 2Ø
```

Explication : ADR est l'adresse d'implantation de notre programme objet, en numérotation décimale : BASIC ne connaît que ce mode de numérotation, tandis que l'assembleur ne connaît que l'hexadécimal.

— Les opérandes, ici nommés A et B sont stockés aux adresses ADR et ADR+1 par POKE.

— Le programme objet est lancé par EXEC ADR+3 ; ce dernier nombre est effectivement l'adresse d'exécution.

— Après retour au BASIC, le résultat est lu dans la case mémoire ADR+2.

— GOTO 2Ø : on boucle pour recommencer...

Il est maintenant possible d'exécuter notre programme autant de fois qu'on le désire ; A et B doivent être des entiers entre Ø et 255 (limites d'un octet). Le résultat est lui-même un entier, dans les mêmes limites ; pas de problème pour les petites valeurs de A et B, mais regardons ces exemples :

A	B	R	
2ØØ	55	255	
2ØØ	56	Ø	
2ØØ	57	1	etc

On dit qu'on a fait une addition « modulo » 256. Il y a bien sûr moyen de repérer le débordement de capacité, c'est un des rôles du registre P. En tout cas, notre programme permet de s'habituer à l'arithmétique un peu spéciale du microprocesseur. Il peut servir à tester toutes les opérations, qu'elles soient arithmétiques (addition, soustraction, multiplication, décalages...), mais aussi logiques (négation, et, ou, ou exclusif...).

Que l'on se reporte au manuel pour leur mnémonique.

Les modes d'adressage

Dans notre premier programme, lorsque nous avons voulu charger un opérande dans l'accumulateur A, nous avons utilisé l'instruction :

LDAA M1

qui a pour signification : « charger dans A le contenu de la mémoire M1 ».

C'est ce qu'on appelle l'adressage direct, parfois l'adressage direct étendu. Plutôt que d'utiliser l'étiquette M1, nous aurions pu écrire :

LDAA $AØØØ, et le code objet aurait été le même : B6 AØØØ.

Il est parfois utile de charger l'accumulateur A, ou un autre registre, avec une *constante*, sans passer par l'intermédiaire d'une case mémoire ; on dit que l'on fait de l'*adressage immédiat* et l'on écrit :

LDAA #$FF traduit par 86FF

Le mnémonique LDAA est le même, mais il ne se traduit pas de la même façon. C'est le # qui signale l'adressage immédiat.

Il n'est pas toujours simple de décider quand l'adressage immédiat est préférable à l'adressage direct : c'est (un peu) la différence entre constantes et variables en BASIC.

Il existe encore deux modes d'adressage pour le 6803 :

— L'adressage direct en page Ø. C'est un adressage direct un peu spécial ; il est utilisé quand la case mémoire dont on veut utiliser le contenu est en « page Ø », c'est-à-dire au tout début de la mémoire : $ØØØØ à $ØØFF ; il suffit alors de donner comme opérande le second octet de l'adresse ; le symbole de cet adressage est < :

LDAA <$F5 codage 96F5

équivaut donc à :

LDAA $ØØF5 codage B6ØØF5

Gain essentiel : un octet... La page Ø contient donc de façon privilégiée les variables très souvent utilisées : c'est ce que fait le BASIC d'Alice. Mais il vaut mieux ne pas y mettre les variables de nos programmes.

— L'adressage indexé. Plus important, ce mode d'adressage est utilisé quand on doit travailler avec une série de données ; il est, en gros, l'analogue des tableaux de variables en BASIC. Donnons le principe :

On commence par mettre dans IX, le registre d'index, la première adresse, celle de la première donnée qui nous intéresse :

LDX #TABLE

Il s'agit d'un adressage immédiat et non direct : on ne veut pas mettre en X le *contenu* de la case mémoire étiquetée TABLE, mais bien la valeur de cette étiquette ; c'est une adresse, donc de longueur 16 bits comme le registre IX.

On peut alors charger dans A le contenu de la case mémoire indexée par IX :

LDAA $Ø,X

Si l'on veut le contenu de la case d'adresse IX+1, dans B :

LDAB $1,X

Plus intéressant, une seule instruction suffit pour incrémenter ou décrémenter IX :

INX ou DEX

On peut ainsi se « promener » dans la table des données. Mais pour mettre en oeuvre cet adressage indexé, il nous faut maintenant parler des branchements en langage machine.

Les branchements

Comme en BASIC, un programme en langage machine se déroule normalement dans l'ordre, instruction par instruction. Plus précisément, le registre PC est incrémenté après chaque instruction.

Mais, comme en BASIC, il est possible de rompre cette logique, et ce de trois façons différentes :

— par un branchement inconditionnel,
— par un appel à un sous-programme,
— par un branchement conditionnel.

• Les branchements inconditionnels, analogues donc au GOTO du BASIC. En assembleur, deux possibilités :

JMP adresse (comme JUMP, saut)

ou bien

BRA déplacement ou étiquette.

Dans le premier cas, pas de problème, c'est l'analogue du GOTO. Le second type de branchement se nomme branchement relatif : le registre PC est augmenté du déplacement indiqué, ce déplacement contenant dans un octet :

— si la valeur de cet octet est entre Ø et 127, le déplacement se fait vers l'avant;

— si la valeur de cet octet est entre 128 et 255, le déplacement se fait vers l'arrière.

Ce qui importe pour nous, c'est que l'assembleur calcule automatiquement ce déplacement : il suffit d'étiqueter l'endroit où on veut aller :

```
        BRA SUITE
        NOP
        NOP (deux instructions qui ne font rien. No operation)
SUITE....
```

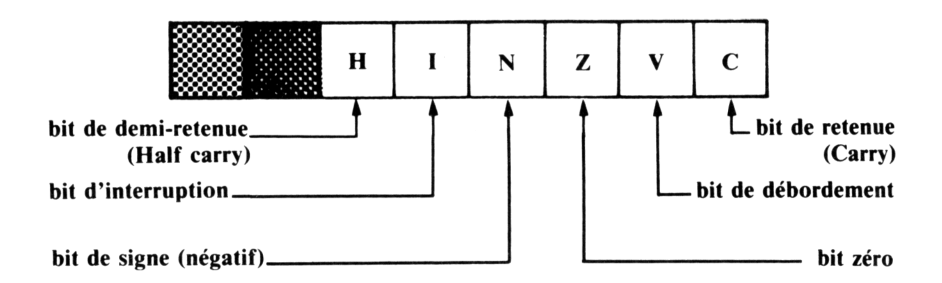
L'assemblage donnera :

2ØØ2 pour la traduction de BRA SUITE, le déplacement a bien été calculé : 2Ø est le code de BRA, Ø2 le déplacement.

• Les appels aux sous-programmes se font comme en BASIC :

JSR suivi d'une adresse ou étiquette (jump to subroutine)
et RTS pour le retour au programme appelant.

• Les branchements conditionnels. Quelques précisions sur le registre d'état P; voici sa description bit par bit :



Le détail des instructions qui affectent le registre P est dans le manuel; donnons cependant quelques précisions;

— Le bit Z est mis à 1 quand le résultat d'une opération ou d'un transfert est nul. Il est également mis à 1 quand le résultat d'un test (TST, BIT ou CMP..) est «oui».

Après chacune de ces instructions, on peut donc faire les branchements conditionnels suivants :

BEQ déplacement (branchement si «equal» à Ø)

BNE déplacement (branchement si non égal)

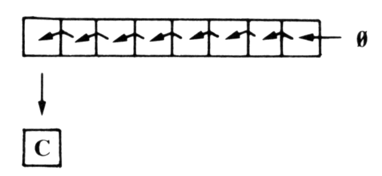
Dans le premier cas, le branchement ne se fait que si l'indicateur d'état P a décelé un nombre nul. C'est le contraire dans le second cas.

— Le bit N est mis à 1 quand le résultat d'une opération ou d'un transfert est négatif : plus précisément, est considérée comme négative une donnée dont l'écriture binaire sur 8 bits commence par 1 ; pour nous, il s'agit d'un entier entre 128 et 255. Les instructions de branchements s'écrivent alors :

BMI déplacement (branchement si moins)

et BPL déplacement (branchement si plus)

— Terminons avec le bit C. Il est mis à 1 lorsqu'il y a une retenue, mais il est aussi utile lors des instructions de décalage ou de rotation; par exemple, lors d'une instruction ASL, les bits se déplacent de la façon suivante :



Cette opération est importante, car elle correspond à une multiplication par deux : on voit que C joue encore un rôle de retenue éventuelle. Les branchements conditionnels sont en ce cas :

BCC et BCS

Nous n'insisterons pas plus, pour les débutants, c'est surtout le bit Z qui sera utilisé.

Un programme musical

Utilisons toutes ces notions pour écrire un programme qui joue une partition musicale. Voici le programme source :

```
        ORG     $AØØØ
        EXC     DEBUT
DEBUT   LDX     #TABLE
BOUCL   LDAB    #TEMPO
        LDAA    $Ø,X
        BEQ     FIN
        PSHX
        JSR     ROUT
        PULX
        INX
        BRA     BOUCL
FIN     RTS
ROUT    =       $FFAB
TEMPO   =       $2
TABLE   DFO     $1Ø
        DFO     $24
        ....
        DFO     $8Ø
        DFO     $ØØ
```

Quelques explications : un son est émis en appelant le sous-programme d'adresse $FFAB : l'accumulateur A doit contenir la fréquence, B la durée. Nous avons, pour plus de lisibilité, utilisé des *symboles* ROUT et TEMPO, qui sont initialisés par la directive = (attention de la faire encadrer de blancs).

Deux instructions nouvelles :

PSHX, PULX

Ce sont des instructions d'empilement et de dépilement. Il faut savoir en effet que la routine qui joue une note *modifie* les registres ; pas de problème pour A et B qui sont réinitialisés au début de chaque boucle, mais c'est plus ennuyeux pour IX, qui joue le rôle d'un index pointant sur chaque note ; avant chaque appel de la routine, nous allons *empiler* la valeur de IX par PSHX, puis la *dépiler*

au retour par PULX. Il ne reste qu'a compléter la table de fréquences, en hexadécimal, en terminant par Ø pour que puisse se faire le branchement sur la fin. Ce programme est appelé très simplement par BASIC :

EXEC 4Ø96Ø

Le clavier-piano-ASCII

Voici un programme qui est un peu plus interactif que le précédent. Il s'agit de transformer le clavier d'Alice en clavier musical.

```
;PIANO
;*****

        ORG     $AØØØ
        EXC     DEBUT
DEBUT   LDAB    #TEMPO
        JSR     $F883
        CMPA    #$Ø
        BEQ     DEBUT
        CMPA    #$3
        BEQ     FIN
        JSR     $FFAB
        BRA     DEBUT
FIN     RTS
TEMPO   =       $1
```

Il n'y a pas d'instruction vraiment nouvelle. On utilise ici une autre routine du BASIC , celle qui scrute le clavier; ce sous-programme d'adresse $F883 renvoie dans l'accumulateur A le code ASCII de la touche du clavier qui a été activée. Si aucune touche n'a été pressée, il renvoie Ø : on doit donc rappeler la routine tant que A est nul (BEQ DEBUT). Nous avons décidé d'interrompre le programme si la touche activée est BREAK, de code 3. Sinon, on se branche sur la routine du programme précédent; A n'a pas été modifié... ce qui fait que l'on joue une note de fréquence le code ASCII de la touche.

Il est bien sûr plus intéressant, du point de vue musical, de transformer le clavier en *vrai* piano. On peut s'en sortir, par exemple, en ajoutant une cascade de comparaisons CMPA... pour tester les touches ; il suffit alors de charger la fréquence désirée dans A avant d'appeler la routine musicale : l'adressage indexé pourrait ici être très utile.

Un dernier exemple musical :

```
        ORG     $A∅∅∅
        EXC     DEBUT
DEBUT   LDAB    #TEMPO
        CLRA
BRUIT   LDX     FREQ
        EORA    #$8∅
        STAA    $BFFF
BOUCL   DEX
        BNE     BOUCL
        DECB
        BNE     BRUIT
        RTS
FREQ    DFD     $FFFF
TEMPO   DFO     $3∅
```

Ce programme n'utilise plus la routine musicale ; le principe est le suivant : pour exciter le haut-parleur, il faut « titiller » le 8ème bit (le plus à gauche) de la case mémoire $BFFF : cela se fait en mettant dans A la valeur $∅ (par CLRA), puis en faisiant basculer ce bit de 1 à ∅ ou de ∅ à 1 par EORA # $80

IX n'est pas ici utilisé comme index mais comme compteur de temps entre deux excitations.
Le registre B joue le rôle d'un indice dans une boucle de répétition. Pour utiliser ce programme, écrivons le programme BASIC suivant :

```
5   ADR=4∅96∅
1∅  INPUT "X1,X2,B" ;X1,X2,B
2∅  POKE ADR+19,X1
3∅  POKE ADR+2∅,X2
```

```
40  POKE ADR+21,B
50  EXEC ADR
60  GOTO 10
```

X1 et X2 seront respectivement les poids fort et poids faible de IX, c'est-à-dire que :

IX = X1 * 256 + X2

Plus IX est petit, plus le son sera aigü ; tous les essais sont possibles.

Un dernier exemple, qui pourrait être le début d'un programme de jeu vidéo ; l'important, c'est l'utilisation du semi-graphique :

```
          ORG     $A000
          EXC     DEBUT
RAQU      DFO     $A3
POSX      DFO     $14
POSY      DFO     $17
DEBUT     JSR     DESS
          LDAA    POSX
          DECA
          STAA    POSX
          JSR     DESS
          RTS
DESS      LDAA    RAQU
          LDAB    POSX
          STAB    $3281
          LDAB    POSY
          STAB    $3280
          JSR     $F9C6
          RTS
```

Le sous-programme DESS charge dans A le code ASCII du caractère à afficher, ici un caractère semi-graphique. Puis sont mises dans les cases mémoire $3281 et $3280 la colonne et la ligne où l'on veut cet affichage ; le programme d'affichage est alors appelé. Il est à l'adresse $F9C6.

Le programme principal se contente de dessiner les deux moitiés de la raquette : beaucoup de travail encore avant qu'elle ne se déplace...

UTILISATION



UN MICRO-ORDINATEUR CHEZ SOI, POUR QUOI FAIRE ?

L'informatique.est dans l'air du temps. Elle intrigue, elle fascine, elle inquiète...

Pourquoi avez-vous acheté un micro-ordinateur, pourquoi songez-vous à en achetez un ? Peut-être ne trouvez-vous pas très agréable de vous sentir tenu à l'écart d'un phénomène technique, sociologique et culturel aussi important. Ces raisons-là en valent bien d'autres.

La vraie révolution informatique, celle que George Orwell n'avait pas prévue, celle que personne ne pouvait prévoir avant 1978, c'est celle de la micro-informatique, c'est-à-dire la révolution de la décentralisation, celle de l'éclatement.

Aujourd'hui, pour moins de trois mille francs, vous pouvez faire entrer un micro-ordinateur dans votre foyer ? Vous en ferez ce que vous voudrez, mais quoi ?

Principalement, le micro-ordinateur vous familiarisera avec l'informatique, c'est-à-dire avec une méthode de traitement de l'information, une technique de communication que, seule, la pratique permet de comprendre et de maîtriser.

Il se pourrait donc bien qu'un jour, mémoire, système d'exploitation, langage, assembleur, BASIC, interface, microprocesseur, modem, n'aient plus de secrets pour vous : vous le devrez à cette petite boîte rouge.

Bien sûr, l'informatique n'est pas une religion et même si son utilité immédiate est celle d'une initiation, chacun attend d'elle qu'elle rende un service.

Les service rendus par la micro-informatique dans le monde professionnel sont aujourd'hui assez bien déterminés. Il existe même trois catégories de progiciels, traitement de texte, tableau et base de données autour desquelles s'articulent toutes les applications bureautiques.

En revanche, les domaines d'application de la micro-informatique dans le cadre familial, sont, aujourd'hui encore, entourées d'un halo plus ou moins artistique qui révèle la perplexité des fabricants de matériel, comme de logiciel.

Ce chapitre vous donne un apperçu de ce qu'il est possible de faire avec un micro-ordinateur personnel.

Nous n'avons pas cherché à être complet, mais à faire un tour d'horizon aussi large que possible en dégageant trois domaines d'application privilégiés :

— utilitaire,
— éducatif,
— jeu.

Dans chacun d'eux, nous dirons ce qui a déjà été fait et ce qui peut l'être. Mais il reste le champ des applications inexplorées que nous nous plaisons à imaginer fort vaste. Car un micro-ordinateur, c'est aussi une machine à rêver...

DES PROGRAMMES UTILES

Il s'agira ici de programmes utiles à la vie familiale telle qu'elle est aujourd'hui. Dans ce domaine, la vraie révolution ne viendra pas de la micro-informatique (un micro-ordinateur n'est pas capable de cirer vos chaussures ou de repasser vos chemises) mais de la robotique. En attendant, certaines qualités des micro-ordinateurs sont exploitables dans la vie domestique.

L'ordinateur est un outil de gestion très efficace : faire un bilan, tenir à jour des stocks, un fichier, une comptabilité sont des opérations que l'informatique facilite considérablement.

Si vous êtes de ceux qui organisent leur vie quotidienne assez méticuleusement, vous pourrez faire appel au micro-ordinateur pour :

— gérer le budget familial : nature et volume des dépenses, état des comptes, planification de vos investissements, prévision en bourse;

— mettre à jour votre agenda et rendre automatique la recherche d'un nom ;

— mettre en fiches votre bibliothèque, vos disques, vos recettes de cuisine ou votre collection de timbres pour retrouver en quelques secondes ce que vous avez archivé ;

— calculer vos impôts, vos frais professionnels ;

— prévoir l'emprunt le plus avantageux pour l'achat d'un appartement, d'une automobile, d'un bateau en fonction de vos possibilités financières.

Pour la réalisation de programmes effectuant ces différentes tâches, vous devrez établir au préalable un « cahier des charges » et préciser la nature de vos exigences.

En tenant compte de vos qualités de programmeur et du temps dont vous disposez, vous pouvez décider soit de l'écrire vous-même, soit de rechercher dans le commerce le logiciel le plus adapté à votre problème et compatible avec votre machine.

Les fichiers

Même si le programme écrit par vous pour vos propres besoins est forcément le plus personnalisé et donc plus adapté, les fichiers sont des logiciels longs à développer si l'on veut obtenir un outil de travail efficace et fiable.

Les produits professionnels sont difficiles à concurrencer sérieusement. Ils sont là, il serait dommage de ne pas en profiter.

Signalons aussi que l'achat d'un lecteur de disquettes peut s'avérer indispensable : le stockage et la consultation de données sur bande magnétique (accès séquentiel) sont beaucoup plus fastidieux que sur disquettes (accès direct). D'ailleurs, si l'on veut traiter un volume convenable d'informations, les disquettes s'avèrent indispensables.

Le traitement de texte

Dans les applications professionnelles, le traitement de texte est l'un des logiciels les plus utilisés. De quoi s'agit-il? Pour taper une lettre, rédiger un contrat, le traitement de texte permet de saisir le texte sur un support magnétique avant de l'imprimer sur papier.

Le traitement de texte facilite la correction et la modification. Il est très simple d'enlever, d'ajouter ou de déplacer n'importe où dans le texte et à tout moment, une lettre, un mot, une phrase, un paragraphe entier. On fait imprimer le résultat et s'il faut encore modifier, il suffira de rappeler le texte sur son support magnétique, de procéder à la modification puis de réimprimer.

Les secrétaires auront vite fait de comprendre l'intérêt d'un tel outil : avec une machine à écrire ordinaire, s'il faut ajouter une phrase dans un texte, c'est toute la page qu'il faut retaper; avec un traitement de texte on ne retape que la phrase à insérer.

Voilà pour le bureau. Mais à quoi peut bien servir un traitement de texte à la maison?

On pourrait répondre par une boutade : si vous savez écrire, il vous secondera, si vous ne savez pas, il ne vous servira à rien.

De plus en plus d'écrivains, de journalistes écrivent sur traitement de texte. Les éditeurs et rédacteurs en chef s'en réjouissent.

Encore un usage professionnel direz-vous?

Mais pour un collégien, un lycéen, un étudiant, rédiger une dissertation sur traitement de texte évitera les ratures du brouillon et donnera une présentation très lisible. Ce dont le professeur ne saurait que se réjouir.

D'autre part, l'utilisation d'un traitement de texte bouleverse complètement les habitudes d'écriture. Des expériences faites avec des écoliers ont révélé que certains blocages face à la rédaction manuscrite disparaissaient. Plus de brouillon puis de version recopiée d'après le brouillon. C'est la version définitive qui s'élabore progressivement à partir du brouillon.

Les enfants qui se découragent d'avoir à raturer, supprimer, rajouter, changer les mots, corriger les fautes sont stimulés par le traitement de texte qui leur présente toujours une version propre de leur travail. Ils peuvent la modifier sans en altérer la forme.

Grâce à l'éditeur-assembleur d'Alice, vous pouvez d'ores et déjà vous initier aux principes et à la pratique du traitement de texte. Associé à l'imprimante thermique, vous pouvez produire quelques notes de service familiales du plus bel effet...

Pour les circulaires plus sérieuses, il faudra attendre un vrai logiciel de traitement de texte dédié à Alice et une imprimante de meilleure qualité.

DES PROGRAMMES EDUCATIFS

Alice est un micro-ordinateur dont la simplicité d'accès exerce un pouvoir attractif voulu sur les enfants. Tout naturellement, les applications éducatives sont de celles que les producteurs de logiciels privilégient.

L'emploi d'un micro-ordinateur comme aide à l'apprentissage est récemment apparu comme pouvant être considérable. Dans bien des cas, l'achat d'une machine familiale est motivé par la présence d'enfants.

Mais il est tout aussi vrai que les motivations des parents et des enfants ne coïncident pas toujours. Les enfants voient dans l'ordinateur un jouet sophistiqué, les parents un outil éducatif. Commençons par l'éducatif.

Le micro-ordinateur peut être un outil éducatif à double titre :

— apprentissage de la programmation,

— pratique de logiciels scolaires (didacticiels).

Apprendre avec BASIC

La pratique d'un langage de programmation développe les qualités intellectuelles et forme le caractère.
L'apprentissage du BASIC réclame et développe :

— rigueur,

— clarté,

— esprit d'analyse,

— imagination,

— abnégation.

Le mécanisme qui met en oeuvre toutes ces qualités peut se décrire ainsi :

— Je me fixe un projet (imagination).

— Je m'en fais une idée précise (clarté).

— Je le décompose en éléments simples (analyse).

— Je donne les instructions à la machine (rigueur).

— Si ça ne marche pas, je ne me décourage pas et je recommence (abnégation).

Le but étant fixé, le programmeur n'a de cesse d'obtenir le résultat tant désiré. Il sait qu'il peut compter sur l'infinie patience et l'absolue rigueur d'une machine qui ne se lasse ni ne se fatigue jamais. Les joies provoquées par le succès sont à la mesure des souffrances qui les ont précédées.

La programmation est avant tout une activité intellectuelle. A la différence des activités intellectuelles traditionnelles (de type scolaire), elle est auto-émulative : elle provoque l'envie de progresser pour obtenir, expérimentalement, le résultat recherché.

La rigueur de cette logique est d'un type particulier, proche de celle des mathématiques ou de celle qui régit le codage et le décodage rigoureux de langues anciennes comme le latin.

Trop de rigueur diront certains ! C'est vrai.
Il ne faut donc pas abuser de l'informatique. Comme de toute chose...

Infogrammes présente 4 cassettes d'auto-initiation au BASIC : le cube BASIC.

Apprendre avec les didacticiels

Lorsqu'on songe au micro-ordinateur comme outil pédagogique, c'est en fait à cette catégorie de logiciels scolaires, les didacticiels, que l'on songe.

En prenant le risque d'être trop schématique, disons que les activités d'enseignement peuvent se regrouper autour de deux pôles :

— apprendre (le cours),

— pratiquer (les exercices).

Apprendre, c'est le travail d'un maître, d'un professeur, d'un initiateur. Pratiquer, c'est celui d'un répétiteur.

Il faut bien admettre que le micro-ordinateur se présente comme un excellent répétiteur et comme un piètre maître.

Jusqu'à preuve du contraire, nous nous intéresserons plutôt à ce premier usage d'aide à l'apprentissage.

Dans ce domaine, les modalités d'emploi varient selon les objectifs choisis : transmettre des connaissances, faire acquérir des comportements, former à la méthodologie, compléter le cours habituel (contrôle des connaissances, entraînement méthodique) simuler des situations, des phénomènes.

Remarquons au passage que ce point de vue à au moins le mérite de placer l'enseignant et la machine, non sur des positions concurrentielles mais complémentaires.

L'une des raisons qui fait de l'ordinateur un maître assez médiocre, est la difficulté que l'on éprouve à lire un long texte affiché sur un écran. Les auteurs de didacticiels s'interdisent le plus souvent les longs discours ; ils préfèrent les petits dessins qui valent bien mieux, comme chacun sait.

Voici les logiciels éducatifs nés avec Alice :

EDICIEL

Exercices de calcul
Solfège
Point Bac - Maths 1
Point Bac - Maths 2

INFOGRAMMES

Le moteur à explosion
Ecologie, cycle de vie
L'énergie nucléaire
La géographie mondiale
La France
Le cartable d'Alice (4 cassettes pour l'entrée en 6ème)

VIFI-NATHAN

Alphabet
Lettres en désordre
Lire vite et bien
Multiplications casse-tête
Sauterelle (dans l'esprit de Logo)

DES PROGRAMMES POUR JOUER



Il est probable que pour la grande majorité des enfants comme des adultes, un ordinateur à la maison, ça va d'abord servir à jouer. Il ne faut pas confondre les consoles de jeu qui ne servent qu'à jouer, et les micro-ordinateurs qui peuvent servir à autre chose.

Le jeu a eu l'immense mérite historique de démystifier l'informatique. Il a transformé l'odieux dictateur électronique imaginé par George Orwell dans son roman « 1984 », en un objet familier, quotidien, amusant.

Les premiers logiciels de jeu sur micro-ordinateurs faisaient quasiment tous appel aux réflexes : faire rebondir une balle sur une raquette que le joueur doit déplacer habilement, c'était le scénario d'un des premiers jeux vidéo.

L'inconvénient de ce type de jeux est qu'ils provoquent infailliblement la lassitude et le désintérêt pour un joueur qui à force de pratique est devenu suffisamment habile pour faire durer les parties trop longtemps.

Notons cependant que ce qui apparaît comme un inconvénient pour le joueur, présente un intérêt commercial évident : une fois que vous vous êtes lassé d'un jeu, il vous reste à en acheter un autre...

Progressivement, les caractéristiques de ces jeux se sont diversifiées en même temps que leur nombre grandissait et que leurs qualités graphiques s'affirmaient.

Du point de vue du contenu, on a vu apparaître des jeux qui, outre les réflexes, font appel à d'autres aptitudes : le hasard et la réflexion.

Du point de vue de la forme, les scénarii se sont diversifiés mais ont toujours conservé une place de choix au domaine spatial.

D'autre part, on trouve aujourd'hui des jeux qui peuvent vous opposer, soit à l'ordinateur, soit à un autre joueur humain.

Mais c'est surtout la distribution entre réflexe, chance et réflexion qui permet le mieux d'établir une typologie de ces jeux.

L'un des «tubes» de ces dernières années, le célèbre «Pac Man», doit son succès à un habile dosage entre ces trois aspects du jeu : un peu de hasard, le reste partagé entre stratégie et réflexe.

Jeux disponibles (Décembre 84)

EDICIEL

Jeux de dames
Annexion (jeu d'Othello)
Casse-tête dans le métro
Le sphynx d'or

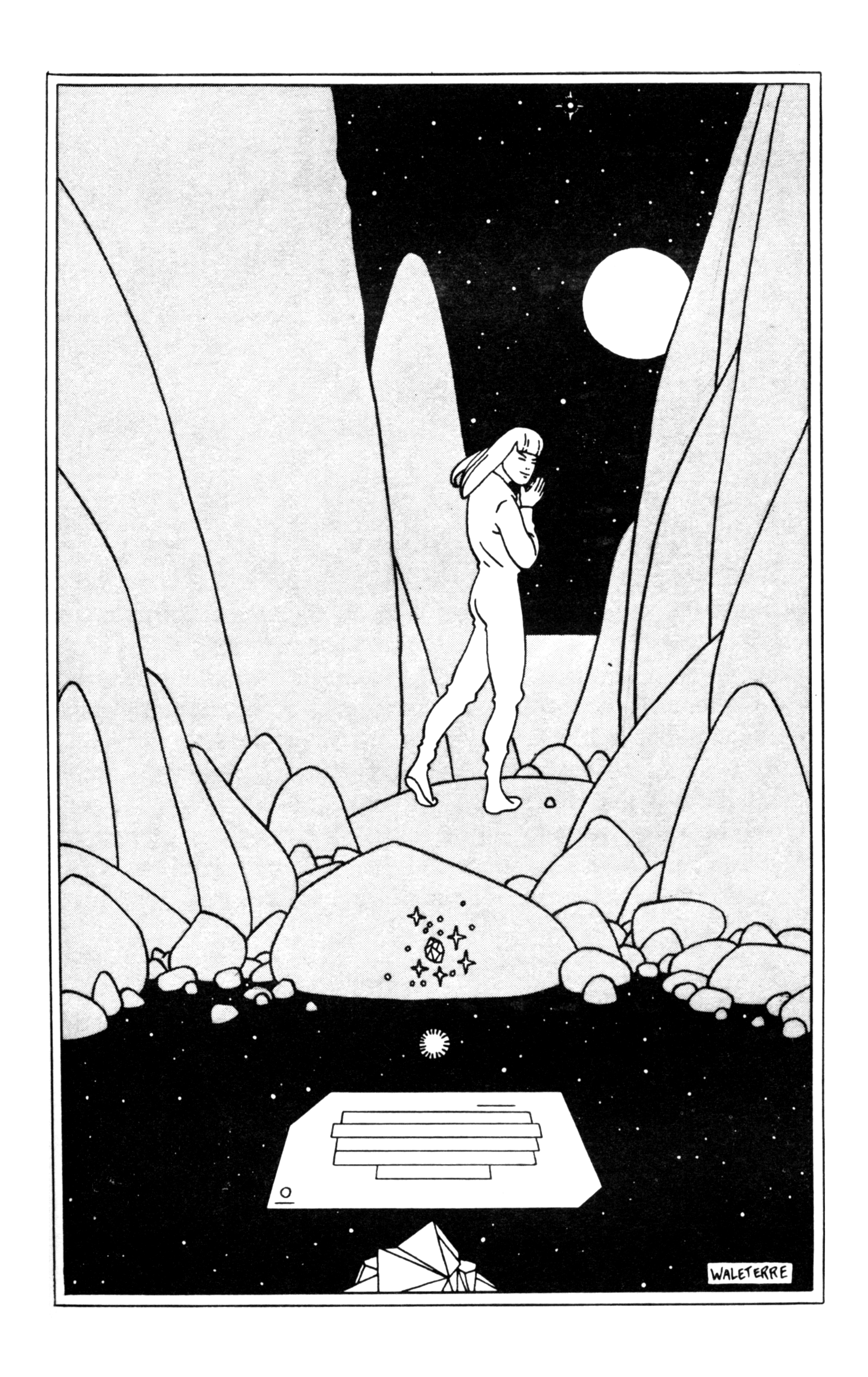
LOGICIELS

Galaxion
La chenille infernale
Bouzy
Crocky
La course aux lettres

VIFI-NATHAN

Solitaire




WALETERRE

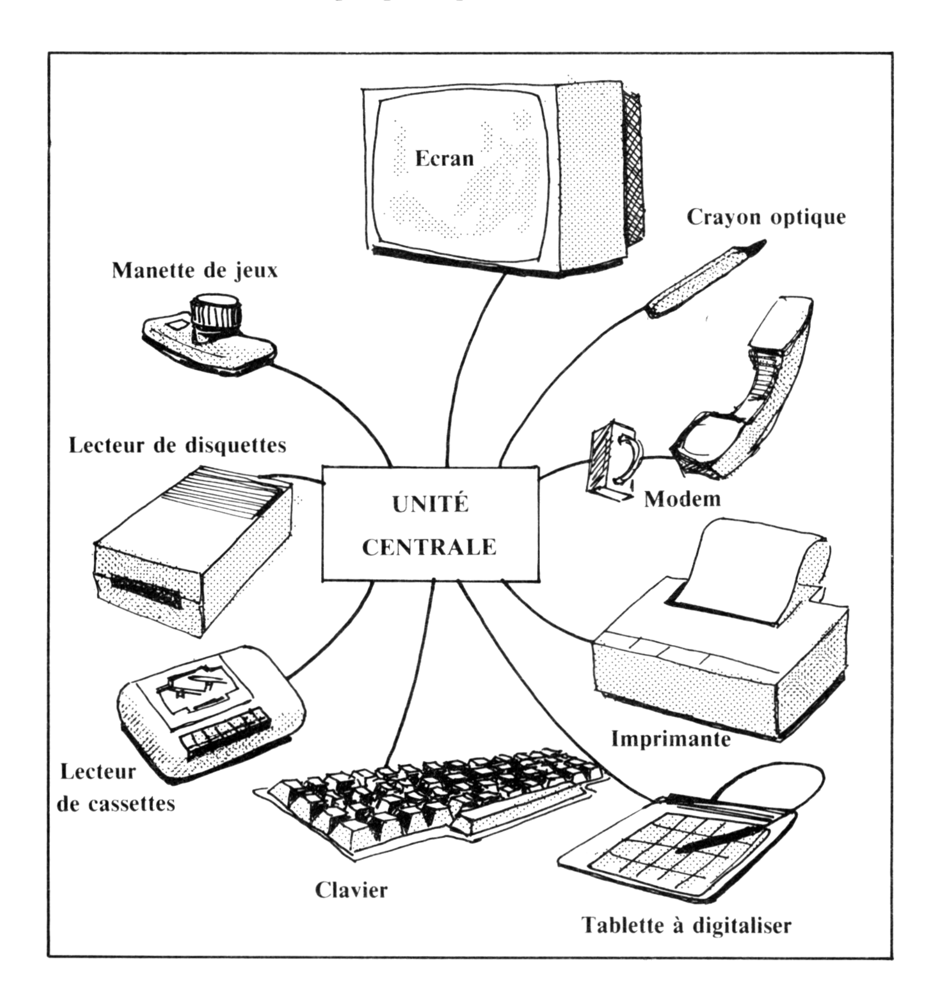
TOUT AUTOUR D'ALICE

La boîte rouge d'Alice n'est pas isolée du monde. La présence sur la carte microprocesseur de *puces d'interfaçage* (voir p. 25) autorise à penser que le problème de la communication fût l'une des préoccupations centrales des concepteurs d'Alice comme de tous les micro-ordinateurs récents. A quoi servirait-il de disposer d'un aussi formidable outil de traitement des informations s'il n'était possible de fournir et recevoir la matière première d'une façon naturelle et aisée ? Mais la communication entre l'utilisateur et le micro-ordinateur se doit de transiter par un appareil, un périphérique qui s'adapte aux possibilités humaines. Les périphériques s'adressent donc aux mains et aux yeux de l'utilisateur selon que l'information rentre ou qu'elle sort...

L'unité centrale, véritable centre nerveux du micro-ordinateur, est entourée de plusieurs périphériques : ce sont des organes de communication indispensables pour sortir l'unité centrale de son profond isolement. Deux de ces périphériques vous sont d'ores et déjà familiers : le clavier et l'écran.

Le clavier : périphérique d'entrée, pour envoyer des informations à l'unité centrale; l'écran : périphérique de sortie, pour afficher en clair des résultats obtenus dans le secret des circuits de l'unité centrale.

Il existe bien d'autres périphériques :



LE LECTEUR-ENREGISTREUR DE CASSETTES

En tant que lecteur, c'est un périphérique d'entrée, en tant qu'enregistreur, c'est un périphérique de sortie.

Le clavier vous est fourni avec la machine. L'écran est absolument indispensable. Le magnétophone le devient rapidement : que ce soit pour profiter d'un logiciel de jeu ou pour «sauver» un programme sur lequel vous avez transpiré quelques heures et qui disparaîtrait définitivement de la mémoire (RAM) au moment où vous couperiez le courant.



Si vous avez acheté Alice version «coffret», pas de problème : le lecteur-enregistreur de programmes vous est alors fourni, ainsi que le câble de raccordement.

Si ce n'est pas le cas, vous allez devoir faire l'acquisition d'un lecteur de cassette. Lequel choisir? A priori, n'importe lequel.

Si vous en avez un, essayez-le, ça devrait marcher; il n'est pas indispensable qu'il y ait une prise pour la télé-commande du moteur, il suffit d'une prise pour l'entrée (souvent notée MIC sur le magnétophone) et d'une prise pour la sortie (prise écouteur).

Le raccordement se fera au moyen d'un câble DIN-5 broches (du côté d'Alice) et DIN-5 broches ou JACK (deux prises mâles) pour le magnétophone. Ces câbles se trouvent sans difficulté dans le commerce.

Une règle générale quand même : il est préférable de choisir un appareil et des cassettes de bonne qualité. S'il y a un «trou» dans la magnétisation d'une bande, ou si le lecteur hocquette une fraction de seconde, alors que rien n'est perceptible pour un enregistrement musical, l'effet peut être catastrophique pour un programme : vous n'obtiendrez rien du tout.

La vitesse de transmission des informations électriques entre l'unité centrale et le magnétophone se mesure en »bauds» (hommage à Emile Baudot, ingénieur français, 1845 - 1903).

En ce qui concerne Alice, elle est de 1 500 bauds, ce qui est assez élevé et a pour conséquence que la lecture ou l'enregistrement des programmes est assez rapide. Un baud correspond à un bit par seconde : sur la bande magnétique, un bit occupe environ 0,03 mm, c'est très peu, et on comprend donc que le magnétophone doit être de bonne qualité pour ne perdre aucun bit.

Lire un programme

C'est la manoeuvre la plus simple. Dans l'ordre :

- Brancher le magnétophone.
- Brancher le micro-ordinateur.
- Placer la cassette en début de bande (retour rapide).
- Ecrire au clavier : CLOAD «.....» (en précisant entre les guillemets le nom du programme cherché) puis taper ENTER.
- Appuyer sur la touche «lecture» du magnétophone.

L'écran s'efface, apparaît la lettre S en haut à gauche (Searching), puis la lettre F (Found) et le nom du programme.

Si l'on ignore ce nom, il suffit d'écrire CLOAD ; sans précision Alice chargera le premier programme rencontré.

Dès l'apparition du message OK, on peut exécuter le programme. Mais il ne faut pas oublier d'arrêter le magnétophone, sinon la cassette continuerait de défiler.

Enregistrer un programme

La manoeuvre est un peu plus délicate; et pourtant, un enregistrement mal fait, et ce sont des heures de travail perdues...

Le magnétophone et le micro-ordinateur étant branchés, on commence par choisir une cassette vierge ou en tout cas une place libre. Attention à bien se positionner *après* la bande amorce.

On écrit ensuite CSAVE «....» en choisissant un nom pour le programme, entre les guillemets.

On met le magnétophone en position d'enregistrement *puis* on appuie sur ENTER; attention à ne pas inverser ces deux manoeuvres, sinon le début de l'enregistrement ne sera pas fait et il sera impossible de relire le programme.

Dés la fin de l'enregistrement, c'est-à-dire lorsque le signal OK apparaît, on arrête le magnétophone; inutile de laisser enregistrer du «blanc».

Ces précautions à prendre sont bien sûr les mêmes dans le cas d'un enregistrement sous l'éditeur ou sous l'assembleur.

Les autres enregistrements

Les instructions que nous venons de décrire concernent les enregistrements de programmes en BASIC. Rappelons qu'il est possible aussi d'enregistrer des *tableaux* avec CSAVE * : il faut alors indiquer le nom du tableau, puis, et entre guillemets, le nom du fichier que l'on veut ainsi créer.

Attention, au moment du chargement par CLOAD * , il faudra redonner le même nom de fichier, mais on pourra changer le nom du tableau; après avoir pris la précaution de le dimensionner correctement. Ce chargement peut être fait à l'intérieur d'un programme, et l'on dispose ainsi d'un traitement de fichiers (rudimentaire).

Conclusion : l'enregistrement des programmes sur cassette est très fiable et relativement rapide. Mais on est quand même loin de l'enregistrement sur disquette : beaucoup plus rapide, il permet un accès *direct* aux informations qui nous intéressent. Sur une cassette, l'accès est dit *séquentiel.* Il est nécessaire de faire défiler la bande jusqu'à l'endroit voulu.

L'IMPRIMANTE



L'imprimante la plus facile à installer pour Alice est celle qui est fournie par le constructeur : c'est une imprimante thermique, qui imprime environ 30 caractères par seconde sur 32 colonnes.

Un avantage de ce type d'imprimantes, outre leur faible prix, c'est le silence. Il existe d'autres types d'imprimantes qui peuvent éventuellment être raccordées à Alice, par la sortie série qui suit la norme RS 232 C.

Imprimantes à aiguilles (les plus répandues)

Les caractères sont formés de points marqués par des aiguilles (picots).

Souvent elles écrivent de gauche à droite et de droite à gauche (bidirectionnelles) et mémorisent une ou plusieurs lignes d'avance.

Inconvénients : elles sont bruyantes et la qualité d'impression est médiocre.

Imprimantes à marguerite

Une roue, telle une fleur des champs, dont les pétales portent à leur extrémité un caractère gravé en relief, tourne devant un rouleau encreur.

Un marteau s'abat sur le caractère sélectionné et provoque l'impression.

La qualité obtenue est dite »qualité courrier», c'est tout dire. Il suffit de changer la marguerite pour accéder à une autre police de caractères.

Inconvénients : un peu lente, un peu chère.

Imprimantes à jet d'encre ou à laser

Dans la technique du jet d'encre, des petites goutelettes (très petites!) électrisées sont envoyées par un gicleur dans un champ magnétique qui les guide vers le papier (système télévision). Il est facile de réaliser ainsi des imprimantes couleurs.

Quand au laser, il permet d'atteindre des vitesses très importantes : par un jeu de miroirs, le rayon imprime un tambour photo-conducteur. Qualité d'écriture excellente, proche de la photo-composition d'imprimerie. Les Japonais annoncent d'ores et déjà des modèles à moins de dix mille francs.

Les instructions à utiliser sont tout simplement :

LLIST pour obtenir la liste d'un programme sur l'imprimante.

LPRINT pour imprimer une expression ou un texte sur l'imprimante. Attention, on ne peut alors utiliser le «@».

Voir également le chapitre sur l'éditeur et l'assembleur, les commandes sont différentes.

AUTRES POSSIBILITES

Il s'agit encore de prévisions, mais certaines semblent très sérieuses.

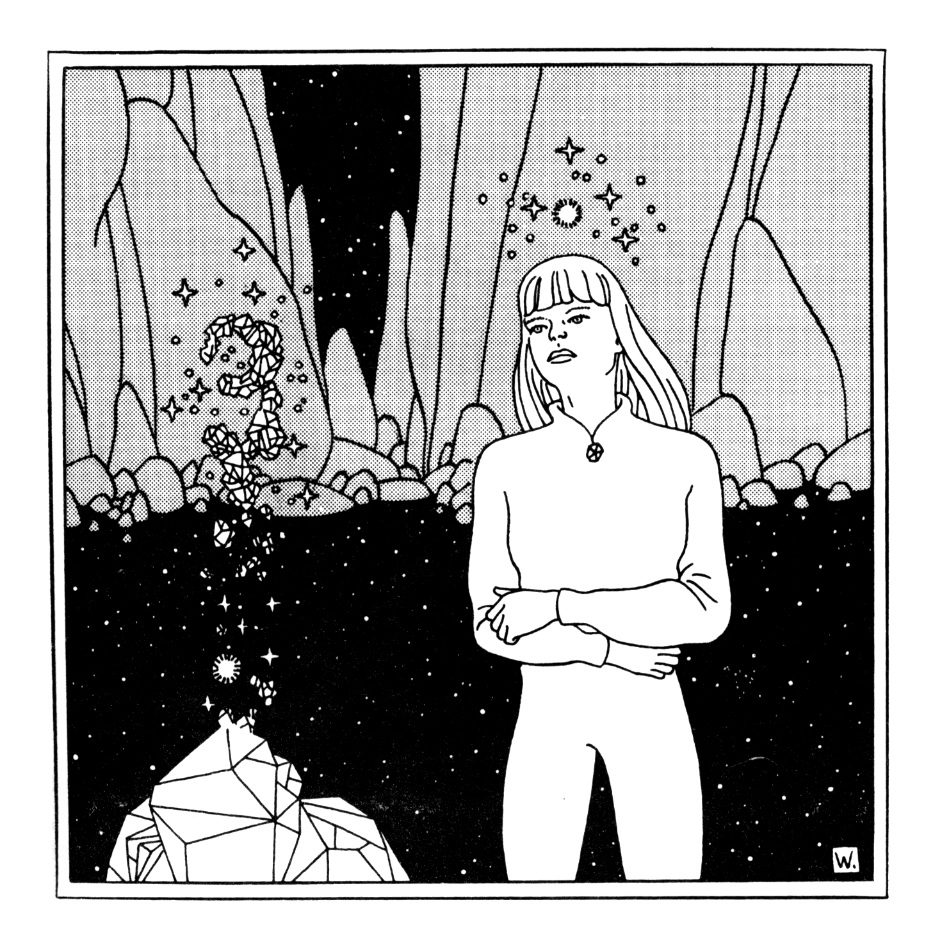
• Alice-90 offre la possibilité d'installer des cartouches de mémoire morte ; la connexion est actuellement cachée par un morceau de plastique : c'est la porte ouverte à des logiciels de jeu très sophistiqués, ou à d'autres langages.

Pourquoi pas un Logo, le langage de la tortue ? Alice risquerait alors de connaître pas mal de succés dans les milieux de l'enseignement.

• Enfin, et cela est moins répandu, Alice-90 permettra de faire de l'incrustation vidéo. De quoi s'agit-il ? Il sera possible de *mélanger* l'image fournie par l'ordinateur et les images d'une des trois chaînes de télévision ; on pourra ainsi réserver 4 lignes au bas de l'écran pour taper un programme, tout en continuant de regarder un feuilleton sur le reste de l'écran... De quoi attraper de solides migraines.

Plus sérieusement, cette incrustation devrait être utilisée au cours d'émissions d'initiation à l'informatique : on pourra faire les exercices sans quitter le «prof» des yeux.

ANNEXE



LES BASES DE LA NUMERATION

Quelques précisions mathématiques

La numération est la branche de l'arithmétique qui envisage les différentes façons d'écrire les nombres entiers. On utilise le plus souvent une numération de position, avec base : la base la plus utilisée est la base 10, et c'est ainsi que le nombre 1024 peut se décomposer :

$1024 = 1 \times 1000 + 0 \times 100 + 2 \times 10 + 4$

Dans le langage courant, 4 est le chiffre des unités, 2 celui des dizaines, 0 celui des centaines et 1 celui des milliers.

On peut encore écrire :

$1024 = 1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$

avec les règles :

$10^3 = 10 \times 10 \times 10 \times 10 = 1000$
$10^2 = 10 \times 10 = 100$
$10^1 = 10$
$10^0 = 1$ par convention

Les coefficients 1, 0, 2, 4 sont les chiffres ; ils doivent tous être plus petits que la base ; on ne dépasse pas 9.

Un nombre entier peut être écrit dans une autre base que la base 10 ; soit B cette base : on mettra B^0, B^1, B^2... à la place de 10^0, 10^1, 10^2... et l'on utilisera des coefficients, ou chiffres, strictement plus petits que B.

Ainsi $49 = 36 + 12 + 1 = 1 \times 6^2 + 2 \times 6^1 + 1 \times 6^0$ permet de dire que 49 s'écrit 121 en base 6.

Les bases les plus utilisées en informatique sont :
— la base deux, ou système binaire,
— la base seize, ou système hexadécimal,
— ainsi que la base dix, bien sûr.

La base deux

Les chiffres ne peuvent être que 0 ou 1. Regardons d'abord comment convertir en base dix un nombre écrit en base deux. Nous utiliserons le préfixe &B pour distinguer les nombres écrits en base deux :

&B 1001 $= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 1 = 9$

Il suffit donc de revenir à la définition. Le plus grand nombre que l'on puisse coder sur un octet est :

&B11111111 $= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255$

Le passage de la base dix à la base deux est moins immédiat. On peut raisonner par soustractions, avec la table suivante :

1	=	2^0	=	&B1
2	=	2^1	=	&B10
4	=	2^2	=	&B100
8	=	2^3	=	&B1000
16	=	2^4	=	&B10000
32	=	2^5	=	&B100000
64	=	2^6	=	&B1000000
128	=	2^7	=	&B10000000

Ainsi $139 = 128 + 11 = 128 + 8 + 3 = 128 + 8 + 2 + 1$

d'où

139 = &B10001011

On peut aussi procéder par divisions successives et écrire le programme BASIC :

```
10  INPUT"N=" ;N
20  B$=""
30  Q=INT(N/2)
40  IF N=2 *Q THEN B$="0"+B$ :GOTO 60
50  B$="1"+B$
60  N=Q :IF N >0 THEN 30
70  PRINT B$
80  GOTO 10
```

La base seize

Le sytème binaire est le plus élémentaire des systèmes de numération : il n'y a que deux chiffres. En contrepartie, l'écriture de nombres tient vite beaucoup de place : huit chiffres pour 128.

Nous allons, pour raccourcir ces écritures, utiliser la base seize ; et parce que $16 = 2^4$, le passage entre les deux systèmes sera aisé ; commençons par expliciter les 16 chiffres ; après 9 on utilise les premières lettres de l'alphabet :

décimal	binaire	hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Nous n'avons pas écrit ici les préfixes :

&B pour un nombre binaire
$ pour un nombre hexadécimal.

La table précédente est à la 6 au point de départ de toutes les conversions : elle donne l'équivalent dans les trois bases d'un demi-octet. C'est pour bien le montrer que nous avons complété par des zéros à gauche les nombres écrits en base deux.

Regardons maintenant comment traiter un octet;

$\&B11010011 = (1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4) + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)16 + 40 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

Ce nombre s'écrira donc avec deux chiffres en base seize :

— le premier chiffre correspond aux quatre premiers bits,
— le second correspond aux quatre derniers.

Ainsi :
&B11010011 = $D3

De la même façon, on peut vérifier que les nombres qui s'écrivent en binaire avec 16 chiffres, c'est-à-dire les entiers inférieurs à 65 536, demandent en hexadécimal 4 chiffres seulement, chacun correspondant à 4 bits.

La conversion hexadécimal-binaire est tout aussi automatique :

$F2 = &B11110010

On peut avoir besoin de convertir un nombre entre les systèmes hexadécimal et décimal sans passer par le binaire : voir par exemple les contraintes du BASIC (nombres en décimal) et de l'assembleur (nombres en hexadécimal).

Dans un sens, c'est facile, on revient à la définition :

$F2 = 15 x 16 + 2 = 242
$8000 = 8 x 16^3 = 32 768

La conversion décimal-hexadécimal est plus délicate : on peut utiliser une table. Par exemple, pour les nombres jusqu'à 255 :

\$10	=	16
\$20	=	32
\$30	=	48
\$40	=	64
\$50	=	80
\$60	=	96
\$70	=	112
\$80	=	128
\$90	=	144
\$A0	=	160
\$B0	=	176
\$C0	=	192
\$D0	=	208
\$E0	=	224
\$F0	=	240

Ainsi, 140 = 128 + 12 = \$80 + \$C = \$8C

On peut aussi écrire le programme BASIC suivant, basé sur des divisions euclidiennes par seize :

```
1Ø  A$="Ø123456789ABCDEF"
2Ø  INPUT "N=" ;N
3Ø  H$=""
4Ø  Q=INT(N/16) :R=N-16 *Q
5Ø  H$=MID$(A$,R+1,1)+H$
6Ø  N=Q :IF N >Ø THEN 4Ø
7Ø  PRINT H$
8Ø  GOTO 2Ø
```

LES MOTS CLES DU BASIC

La mémoire programme

Nous donnons ici la liste de *tous* les mots ou opérations du BASIC, même ceux qui ne sont répertoriés nulle part ailleurs :

FOR	CONT	OFF	SIN
GOTO	LIST	+	COS
GOSUB	CLEAR	−	TAN
REM	NEW	*	PEEK
IF	CLOAD	/	LEN
DATA	CSAVE	↑	STR$
PRINT	LLIST	AND	VAL
ON	LPRINT	OR	ASC
INPUT	SET	>	CHR$
END	RESET	=	LEFT$
NEXT	CLS	<	RIGHT$
DIM	SOUND	SGN	MID$
READ	EXEC	INT	POINT
LET	SKIPF	ABS	VARPTR
RUN	TAB(	USR	INKEY$
RESTORE	TO	RND	MEM
RETURN	THEN	SQR	
STOP	NOT	LOG	
POKE	STEP	EXP	

Les premiers mots correspondent aux commandes et instructions (jusqu'à OFF), puis on trouve les symboles d'opérations et les fonctions.

Certains mots clés semblent n'être là que pour provoquer des erreurs de syntaxe : OFF et USR, bien que le dernier devrait, dans un BASIC ordinaire, permettre de transmettre des paramètres dans un programme en langage machine.

Par contre, VARPTR fonctionne et renvoie l'adresse où est mémorisée une variable ; on peut ainsi constater que les variables numériques sont codées sur cinq octets, un réservé à la mantisse (ou partie avant la virgule), quatre pour les décimales... mais ici encore, la numération binaire joue un grand rôle.

Nous avons donné la liste de mots clés dans un ordre qui n'est pas alphabétique ni, apparemment, logique : il est en fait lié à l'implantation en mémoire des programmes BASIC.

Les programmes BASIC sont enregistrés en mémoire vive à partir de l'adresse 13126 ;

— on trouve d'abord *deux* octets, qui donnent l'adresse à partir de laquelle est enregistrée la ligne suivante du programme. Ainsi, si ces octets ont pour valeur 51 et 88, la ligne suivante commencera à l'adresse 51 x 256 + 88 = 13144 ;

— puis deux octets, qui donnent le numéro de la première ligne du programme BASIC : 0 et 10, c'est la ligne 10 ;

— ensuite, est codée toute la ligne : le texte et les noms de variables sont codés en ASCII, mais les mots clés du BASIC sont codés sur un octet, prenant les valeurs 128 (pour FOR) jusqu'à 200 (pour MEM). Ainsi, une ligne de BASIC est « compressée » ;

— une ligne se termine par un Ø, le programme se termine par Ø,Ø,Ø.

Avec ces quelques indications, il est possible d'intervenir sur un programme BASIC déjà enregistré ; par exemple, il est possible de renuméroter toutes les lignes de 10 en 10...

TABLE DE CONVERSION

D : Décimal

H : Hexadécimal

B : Binaire

D	H	B
1	01	00000001
2	02	00000010
3	03	00000011
4	04	00000100
5	05	00000101
6	06	00000110
7	07	00000111
8	08	00001000
9	09	00001001
10	0A	00001010
11	0B	00001011
12	0C	00001100
13	0D	00001101
14	0E	00001110
15	0F	00001111
16	10	00010000
17	11	00010001
18	12	00010010
19	13	00010011
20	14	00010100
21	15	00010101
22	16	00010110
23	17	00010111
24	18	00011000
25	19	00011001
26	1A	00011010
27	1B	00011011
28	1C	00011100
29	1D	00011101
30	1E	00011110
31	1F	00011111
32	20	00100000
33	21	00100001
34	22	00100010
35	23	00100011
36	24	00100100
37	25	00100101
38	26	00100110
39	27	00100111
40	28	00101000

D	H	B
41	29	00101001
42	2A	00101010
43	2B	00101011
44	2C	00101100
45	2D	00101101
46	2E	00101110
47	2F	00101111
48	30	00110000
49	31	00110001
50	32	00110010
51	33	00110011
52	34	00110100
53	35	00110101
54	36	00110110
55	37	00110111
56	38	00111000
57	39	00111001
58	3A	00111010
59	3B	00111011
60	3C	00111100
61	3D	00111101
62	3E	00111110
63	3F	00111111
64	40	01000000
65	41	01000001
66	42	01000010
67	43	01000011
68	44	01000100
69	45	01000101
70	46	01000110
71	47	01000111
72	48	01001000
73	49	01001001
74	4A	01001010
75	4B	01001011
76	4C	01001100
77	4D	01001101
78	4E	01001110
79	4F	01001111
80	50	01010000
81	51	01010001
82	52	01010010
83	53	01010011
84	54	01010100
85	55	01010101
86	56	01010110
87	57	01010111
88	58	01011000
89	59	01011001

D	H	B
90	5A	01011010
91	5B	01011011
92	5C	01011100
93	5D	01011101
94	5E	01011110
95	5F	01011111
96	60	01100000
97	61	01100001
98	62	01100010
99	63	01100011
100	64	01100100
101	65	01100101
102	66	01100110
103	67	01100111
104	68	01101000
105	69	01101001
106	6A	01101010
107	6B	01101011
108	6C	01101100
109	6D	01101101
110	6E	01101110
111	6F	01101111
112	70	01110000
113	71	01110001
114	72	01110010
115	73	01110011
116	74	01110100
117	75	01110101
118	76	01110110
119	77	01110111
120	78	01111000
121	79	01111001
122	7A	01111010
123	7B	01111011
124	7C	01111100
125	7D	01111101
126	7E	01111110
127	7F	01111111
128	80	10000000
129	81	10000001
130	82	10000010
131	83	10000011
132	84	10000100
133	85	10000101
134	86	10000110
135	87	10000111
136	88	10001000
137	89	10001001
138	8A	10001010
139	8B	10001011
140	8C	10001100
141	8D	10001101
142	8E	10001110
143	8F	10001111
144	90	10010000
145	91	10010001
146	92	10010010
147	93	10010011
148	94	10010100
149	95	10010101
150	96	10010110
151	97	10010111
152	98	10011000
153	99	10011001
154	9A	10011010
155	9B	10011011
156	9C	10011100
157	9D	10011101
158	9E	10011110
159	9F	10011111
160	A0	10100000
161	A1	10100001
162	A2	10100010
163	A3	10100011
164	A4	10100100
165	A5	10100101
166	A6	10100110
167	A7	10100111
168	A8	10101000
169	A9	10101001
170	AA	10101010
171	AB	10101011
172	AC	10101100
173	AD	10101101
174	AE	10101110
175	AF	10101111
176	B0	10110000
177	B1	10110001
178	B2	10110010
179	B3	10110011
180	B4	10110100
181	B5	10110101
182	B6	10110110
183	B7	10110111
184	B8	10111000
185	B9	10111001

D	H	B
186	BA	10111010
187	BB	10111011
188	BC	10111100
189	BD	10111101
190	BE	10111110
191	BF	10111111
192	C0	11000000
193	C1	11000001
194	C2	11000010
195	C3	11000011
196	C4	11000100
197	C5	11000101
198	C6	11000110
199	C7	11000111
200	C8	11001000
201	C9	11001001
202	CA	11001010
203	CB	11001011
204	CC	11001100
205	CD	11001101
206	CE	11001110
207	CF	11001111
208	D0	11010000
209	D1	11010001
210	D2	11010010
211	D3	11010011
212	D4	11010100
213	D5	11010101
214	D6	11010110
215	D7	11010111
216	D8	11011000
217	D9	11011001
218	DA	11011010
219	DB	11011011
220	DC	11011100
221	DD	11011101
222	DE	11011110
223	DF	11011111
224	E0	11100000
225	E1	11100001
226	E2	11100010
227	E3	11100011
228	E4	11100100
229	E5	11100101
230	E6	11100110
231	E7	11100111
232	E8	11101000
233	E9	11101001
234	EA	11101010
235	EB	11101011
236	EC	11101100
237	ED	11101101
238	EE	11101110
239	EF	11101111
240	F0	11110000
241	F1	11110001
242	F2	11110010
243	F3	11110011
244	F4	11110100
245	F5	11110101
246	F6	11110110
247	F7	11110111
248	F8	11111000
249	F9	11111001
250	FA	11111010
251	FB	11111011
252	FC	11111100
253	FD	11111101
254	FE	11111110
255	FF	11111111

L'impression de ce livre
a été réalisée sur les presses
des Imprimeries Aubin
à Poitiers/Ligugé



pour les Éditions Cedic

Achevé d'imprimer en décembre 1984
N° d'impression, L 17485
Dépôt légal, décembre 1984

Imprimé en France

ALICE, ALICE 90, VOTRE MICRO-ORDINATEUR

Jean Delcourt

Le livre qui vous permettra d'installer, comprendre, programmer votre micro-ordinateur.
Vous pourrez jouer, créer, travailler et découvrir l'étendue de toutes les capacités d'Alice et de ses périphériques.





**

291313

ISBN - 2 - 7124 - 1515 - 9


ALICE, ALICE 90, VOTRE MICRO-ORDINATEUR
cedic
nathan